(2)

MULTIGRAPH: AN ARCHITECTURE FOR MODEL-BASED PROGRAMMING

MODEL-BASED PROGRAM SYNTHESIS FOR PARALLEL COMPUTING

PREMOS: PROGRAMMING ENVIRONMENT FOR MODEL-BASED PROGRAM SYNTHESIS

SPC-93154-CMC

VERSION 01.00.00

MARCH 1994

MULTIGRAPH: AN ARCHITECTURE FOR MODEL-BASED PROGRAMMING

Janos Sztipanovits

MODEL-BASED PROGRAM SYNTHESIS FOR PARALLEL COMPUTING

Ben Abbott

PREMOS: PROGRAMMING ENVIRONMENT FOR MODEL-BASED PROGRAM SYNTHESIS

Hubertus Franke

SPC-93154-CMC

VERSION 01.00.00

MARCH 1994

This material is based in part upon work sponsored by the Advanced Research Projects Agency under Grant # MDA972-92-J-1018.. The content does not necessarily reflect the position or the policy of the U.S. Government, and no official endorsement should be inferred.

This document accompanies a videotape of the same presentation recorded live at the Software Productivity Consortium in October 1993. It is recommended that the videotape be viewed with these viewgraphs at hand.

Produced by the
SOFTWARE PRODUCTIVITY CONSORTIUM
under contract to the
VIRGINIA CENTER OF EXCELLENCE
FOR SOFTWARE REUSE AND TECHNOLOGY TRANSFER
SPC Building
2214 Rock Hill Road
Herndon, Virginia 22070

ABSTRACT

MULTIGRAPH: An Architecture for Model-Based Programming by Dr. Janos Sztipanovits, Vanderbilt University
Model-Based Program Synthesis for Parallel Computing by Ben Abbott, Vanderbilt University
PREMOS Programming Environment for Model-Based Program Synthesis by Dr. Hubertus Franke, IBM T.J. Watson Research Center

This presentation consists of three presentations related to model-based software synthesis. The first presentation by Professor Janos Sztipanovits of Vanderbilt University provides an overview of the MULTIGRAPH Architecture (MGA), which is used as a generic framework for model-based programming. Evolution of the architecture has been driven by the requirements of specific applications. Characteristics of these application domains and their impact on the basic design of MGA are discussed.

In the second presentation, Ben Abbott of Vanderbilt University discusses Model-Based Program Synthesis for Parallel Computing. Automatic program synthesis is one of the prime disciplines that can contribute to the advancement of the software engineering of reactive systems. To illustrate, Mr. Abbott presents a large, high-performance parallel instrumentation system used for analysis of turbine engine strain gauge signals produced during altitude testing. The system is called the Computer Assisted Dynamic Data Analysis and Monitoring System (CADDMAS).

In the third presentation Dr. Hubertus Franke of the T.J.Watson Research Center presents Programming Environment for Model-Based Program Synthesis. The development of model-based programming environments is driven by two opposite forces: specialization and standardization. Mr. Franke's presentation addresses the design and implementation of tools which satisfy both above-mentioned forces. In order to overcome this dilemma, the MULTIGRAPH Architecture uses meta-tools.This presentation focuses on the design and implementation of meta-tools and their coordination to form a harmonic environment.

No. 66

MULTIGRAPH: An Architecture for Model-Based Programming

Janos Sztipanovits
Measurement and Computing Systems Laboratory
Vanderbilt University

MEASUREMENT AND COMPUTING SYSTEMS LABORATORY

Research area:
Software technology for embedded computer applications;
Large-scale monitoring, diagnostics and signal processing systems

Personnel:
2 faculty
3 full-time researcher
10 graduate students

Primary sponsors between 1983-1993:
IBM, Boeing, NASA, USA-SDC, OG
USAF-AEDC, Sverdrup Technologies

MILESTONES OF THE MULTIGRAPH RESEARCH

- INTELLIGENT OPERATOR INTERFACE FOR INSTRUMENTATION (MAGNETIC RESONANCE IMAGING, FTIR)
  (1983-1986, IBM)

- FAULT DETECTION, ISOLATION AND RECOVERY IN SPACE SYSTEMS
  (1985- PRESENT, BOEING)
  1993: BOEING STARTED USING MULTIGRAPH IN THE SSF PROGRAM
  1993: THE MULTIGRAPH-BASED DIAGNOSADILITY ANALYSIS TOOL WAS INTRODUCED AND IS BEING TESTED IN SEVERAL BOEING PROGRAMS

- INTELLIGENT PROCESS CONTROL SYSTEM (IPCS) (1986- PRESENT, KRI/OGIS)
  1989: FIRST FIELD TEST IN CO-GENERATOR PLANT
  1992: BETA TESTS IN USA (DU PONT), EUROPE AND JAPAN
  1993: DU PONT PURCHASED THE FIRST COMMERCIAL RELEASE

MILESTONES OF THE MULTIGRAPH RESEARCH

- CADDMAS (1989- PRESENT)
  1990: FIXED PROCESSING 4 CHANNEL SYSTEM
  1992: FLEXIBLE CONFIGURATION 24 CHANNEL PROTOTYPE
  1993: SCALING TO 72 CHANNEL, USE OF ADVANCED DSP ARCHITECTURE ($\sim$100 PROCESSORS, 4GFLOP)

- TRANSIENT DATA PROCESSING (1990-1992)
  1992: WORKING PROTOTYPE ON PARALLEL ARCHITECTURE, FIRST USED IN GE TEST

- IMAGE PROCESSING
  1991-1992: EVALUATE THE USE OF MGA ON NETWORK OF WORKSTATIONS
  1993: WORKING SMALL-SCALE PROTOTYPE ON PARALLEL CADDMAS HARDWARE

- ON-LINE, PARALLEL SIMULATION
  1993: FEASIBILITY STUDY, SMALL-SCALE PROTOTYPE

MODEL-BASED SYSTEMS

MODEL–BASED SYSTEMS DIRECTLY USE MODELS IN THEIR OPERATION.

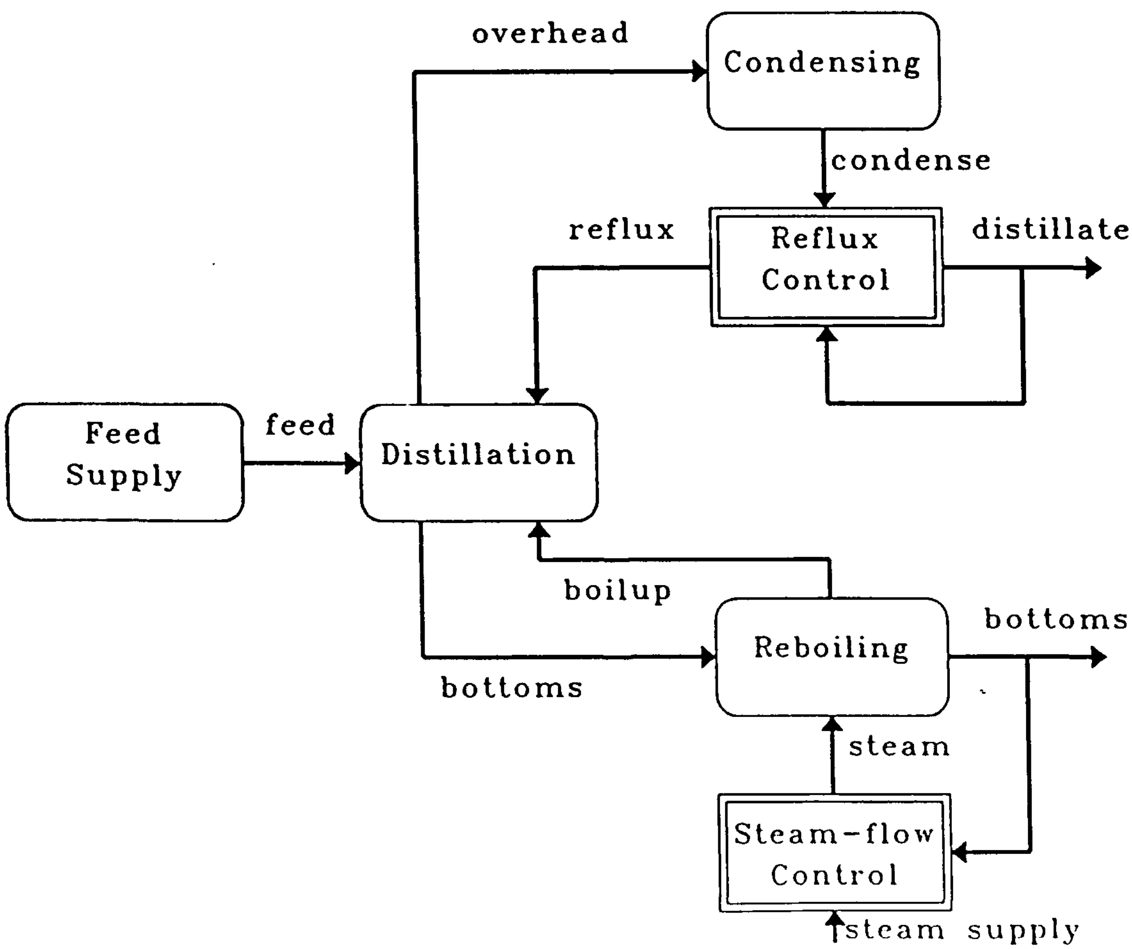
TYPICAL EXAMPLES:

- MODEL-BASED DIAGNOSTICS
- SIMULATION
- MODEL-BASED CONTROL

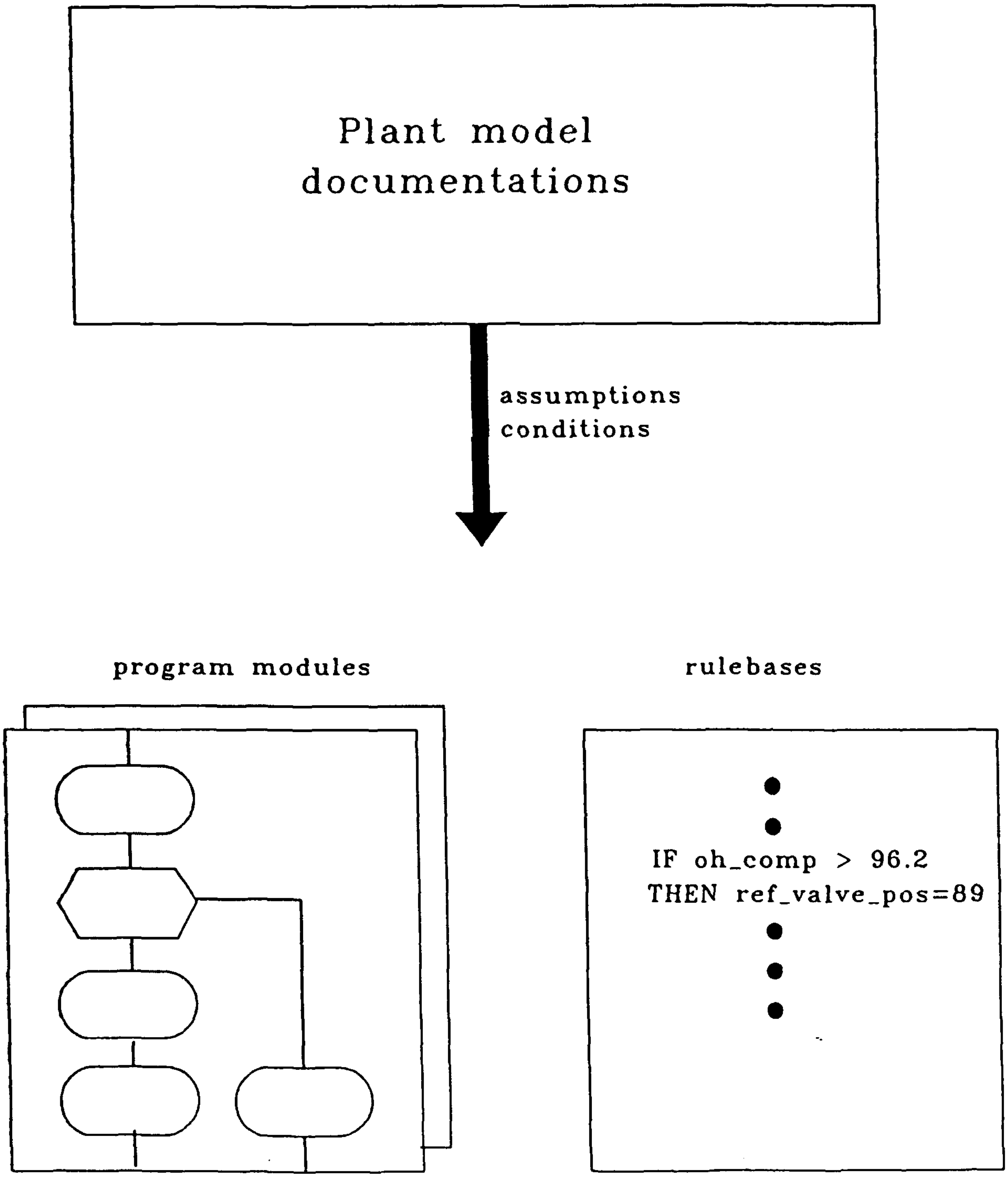
PROGRAMMING ENVIRONMENT

EMBEDDED, REAL-TIME APPLICATIONS (SUCH AS MONITORING, CONTROL, DIAGNOSTIC SYSTMES) CONTINUOUSLY INTERACT WITH PHYSICAL PROCESSES.

THEIR COMPONENTS ARE SELECTED AND DETERMINED BY THE MODELS OF PROCESSES:

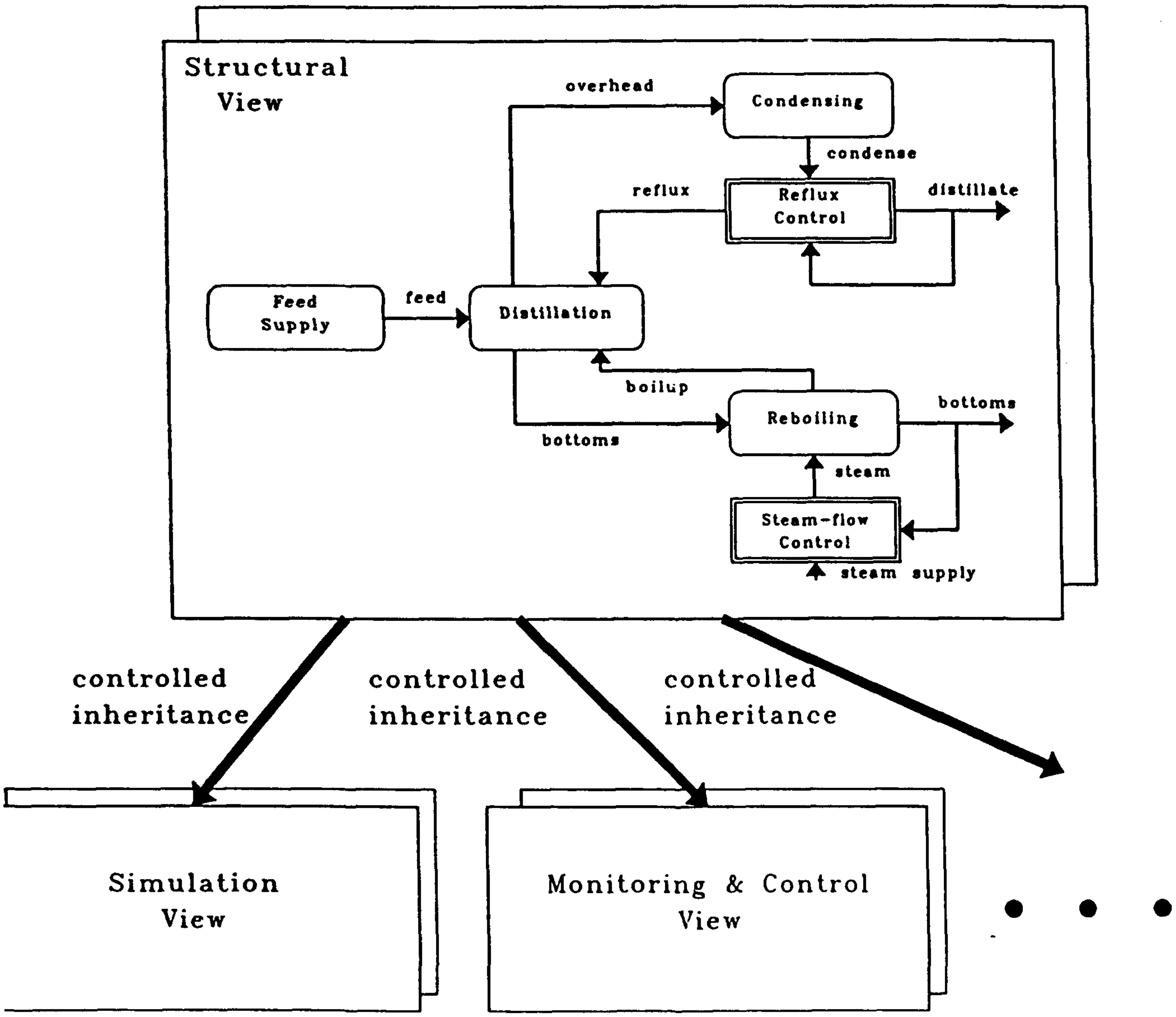


PROGRAMMING ENVIRONMENT
(NOT MODEL-BASED)



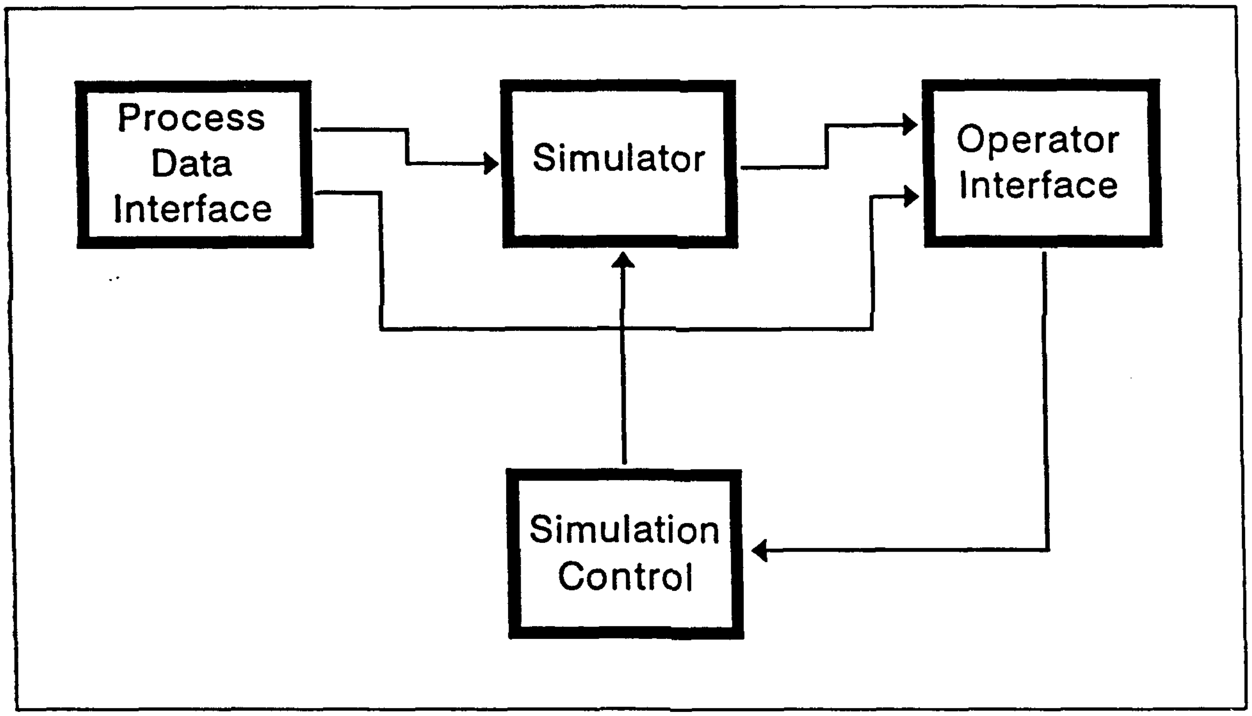
MODEL-BASED PROGRAMMING ENVIRONMENT

;OMPONENTS OF EMBEDDED SYSTEMS \RE DEFINED IN THE CONTEXT OF THE /IODELS OF THEIR ENVIRONMENT:



Example:

"Predicition of catalyst concentration in the mother liquor recycle-loop of the DMT plant."

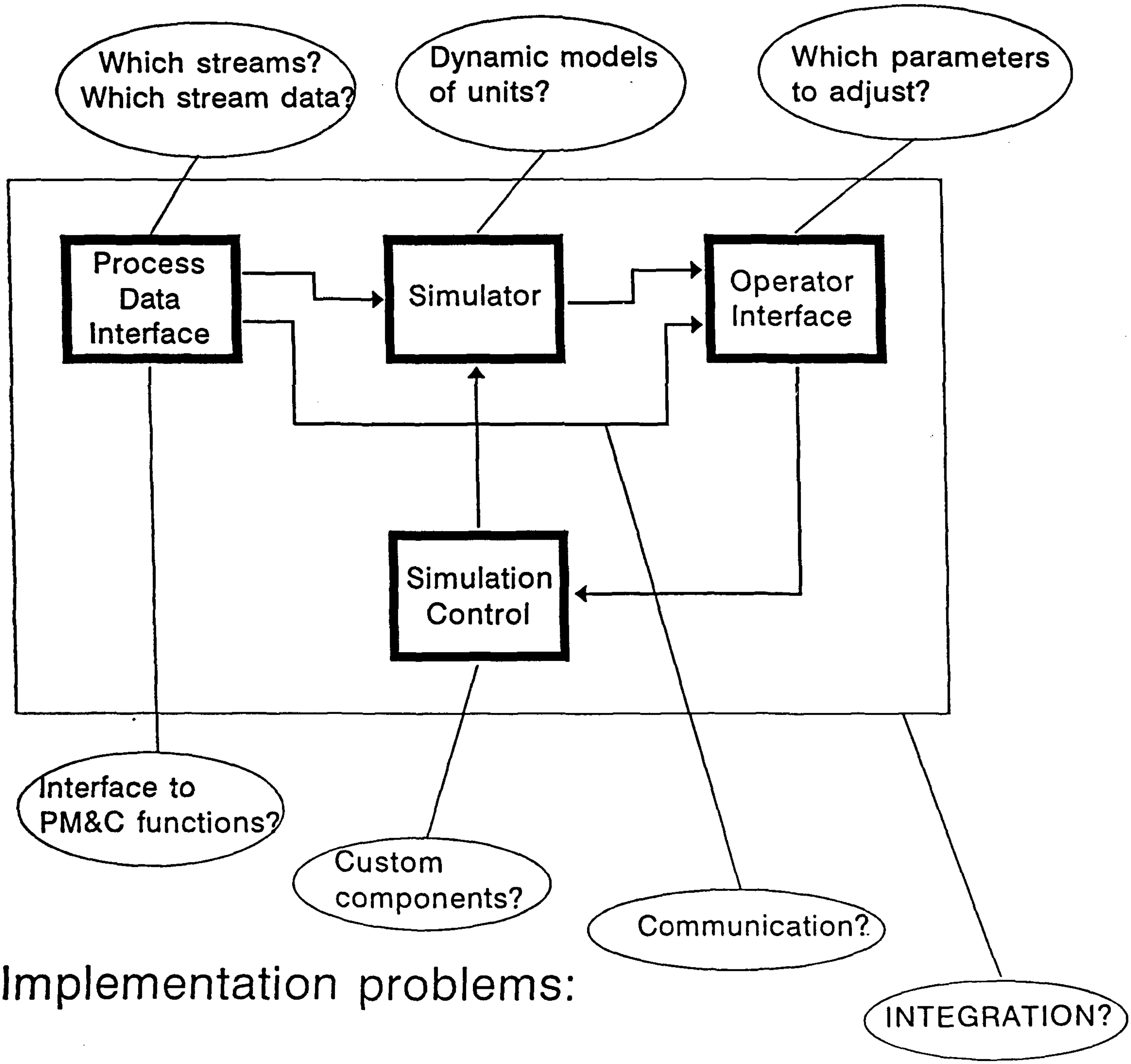


This program includes:

- access to process data
- dynamic simulator
- "customized" human interface
- dedicated simulation control

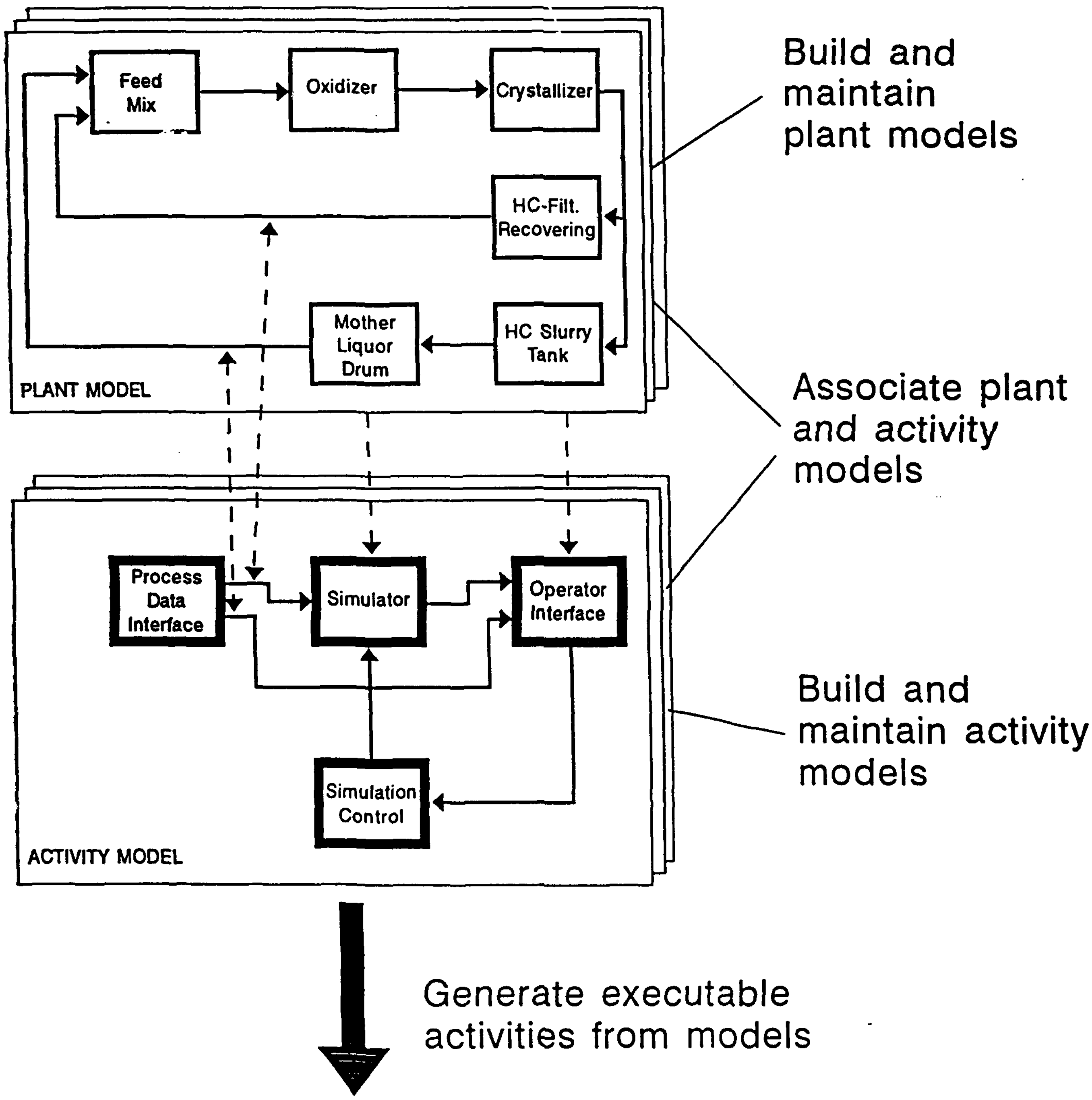
Problems:

Conceptual/logical links to plant information:

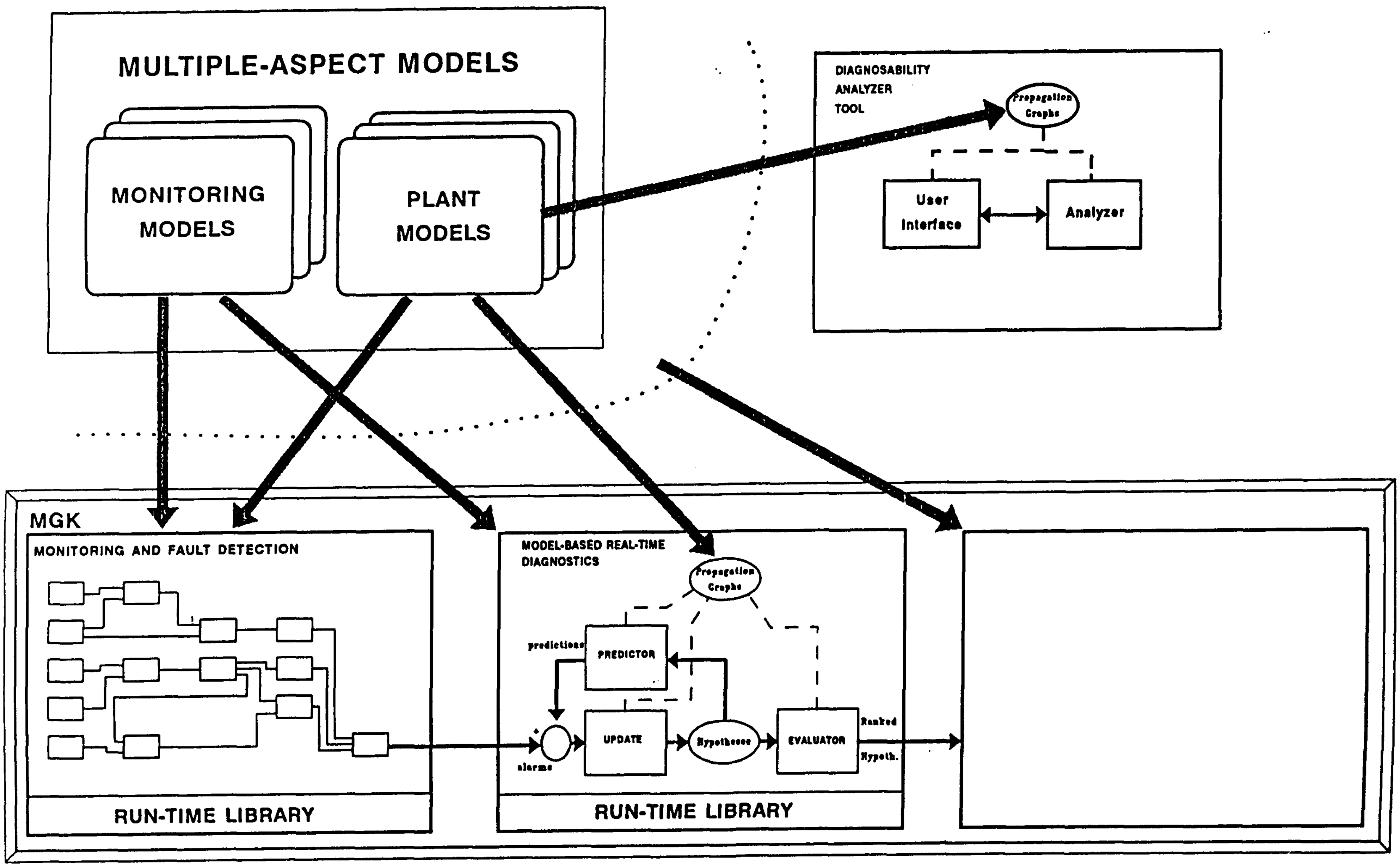


Implementation problems:

Solution with model-based programming environment:



MODEL-BASED PROGRAMMING ENVIRONMENTS AND SYSTEMS



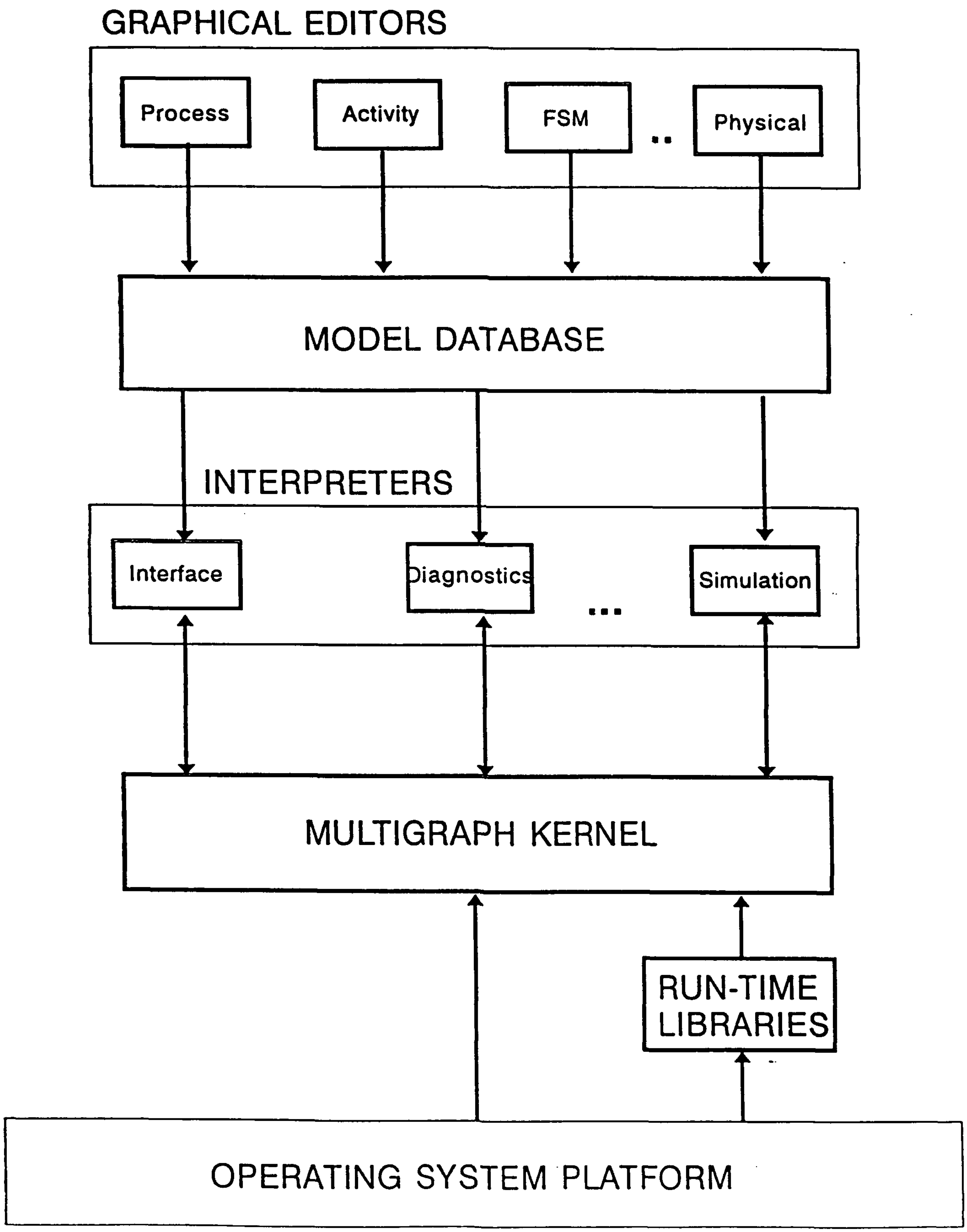
INTERPRETATION OF "MODEL-BASED"

- EXPLICIT REPRESENTATION OF INFORMATION ABOUT THE PLANT, THE COMPONENTS OF THE MONITORING/CONTROL/DIAGNOSTIC SYSTEM AND THEIR RELATIONSHIP

- AUTOMATIC GENERATION OF THE EXECUTABLE CODE FROM THE MODELS

- DIRECT USE OF MODELS IN THE SYSTEM OPERATION

MULTIGRAPH ARCHITECTURE



Model-Based Program Synthesis for Parallel Computing

Ben Abbott
Measurement and Computing Systems Laboratory
Vanderbilt University

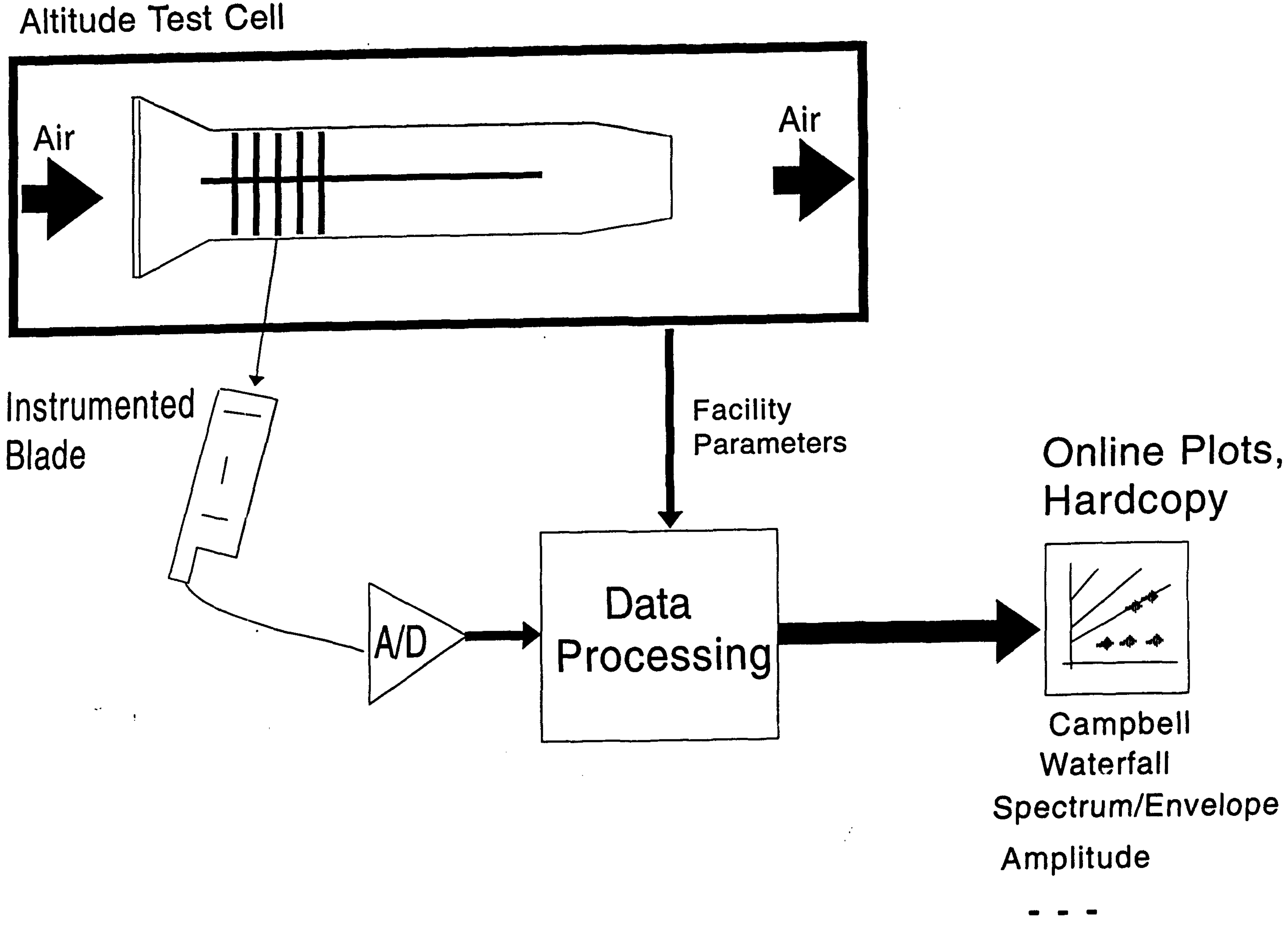
Outline

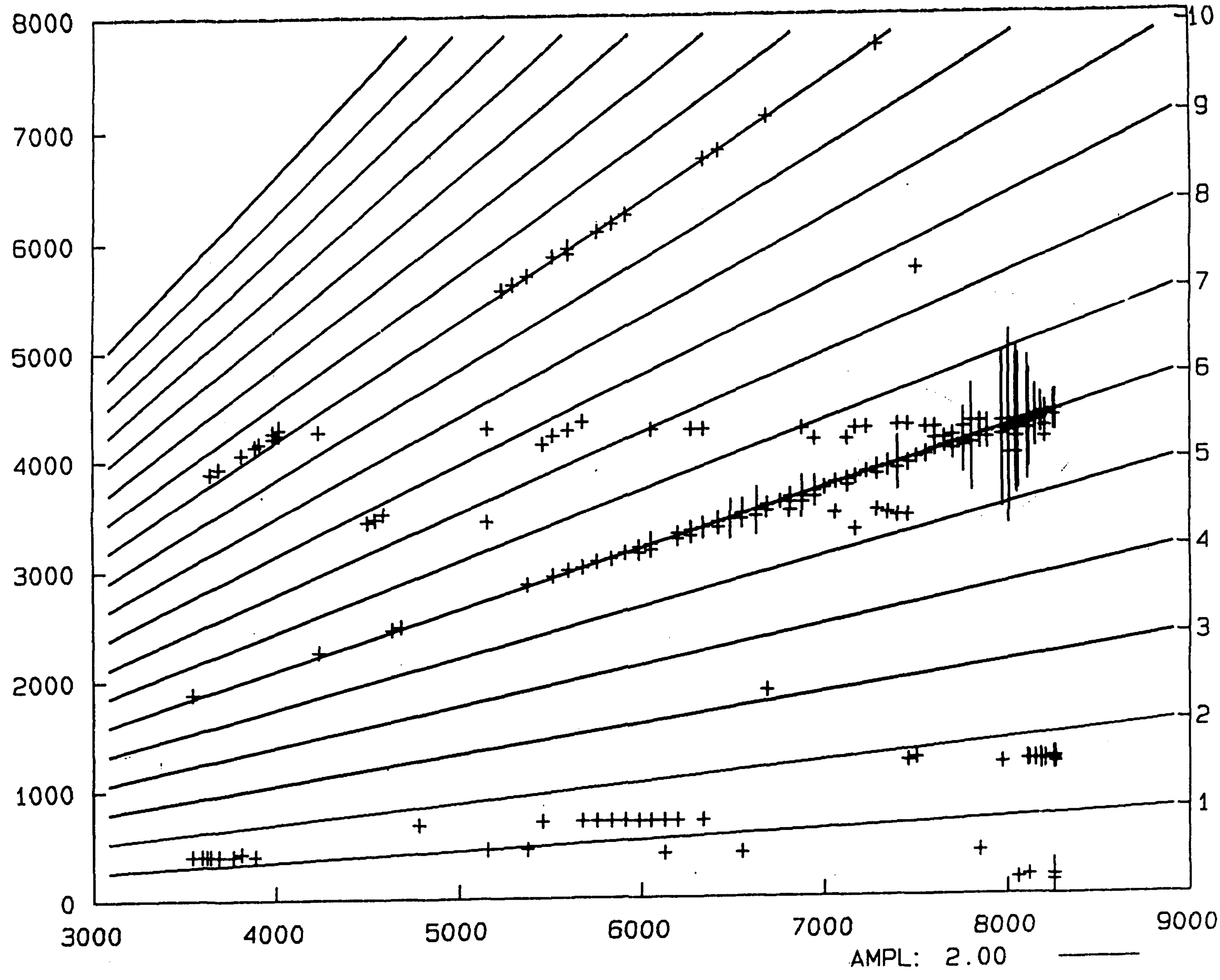
- Motivation
- An Example Problem
- The Model-Based Solution
  (1) Steps taken
  (2) Models
- Conclusion

Motivation

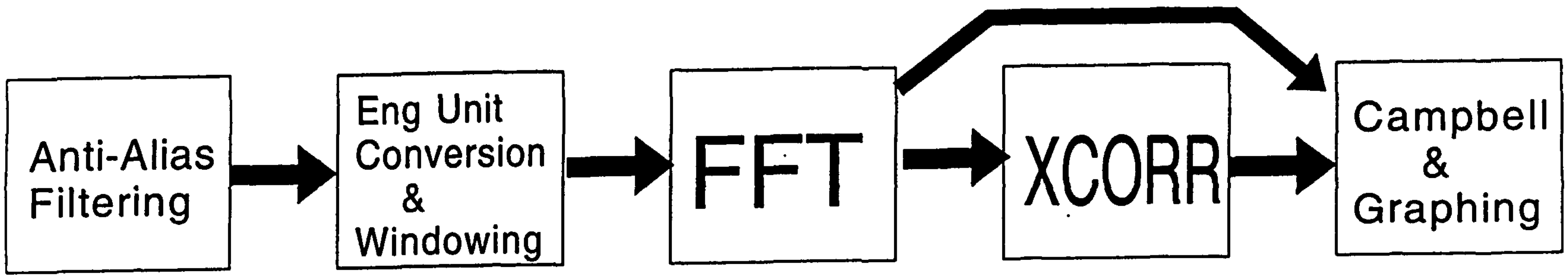
- Large parallel instrumentation systems are needed
- Software is complex
- Hardware configuration and management
- Parallel processing issues:
  - Synchronization
  - Assignment
  - Loading
- Bugs propagate

Turbine Engine Stress Testing



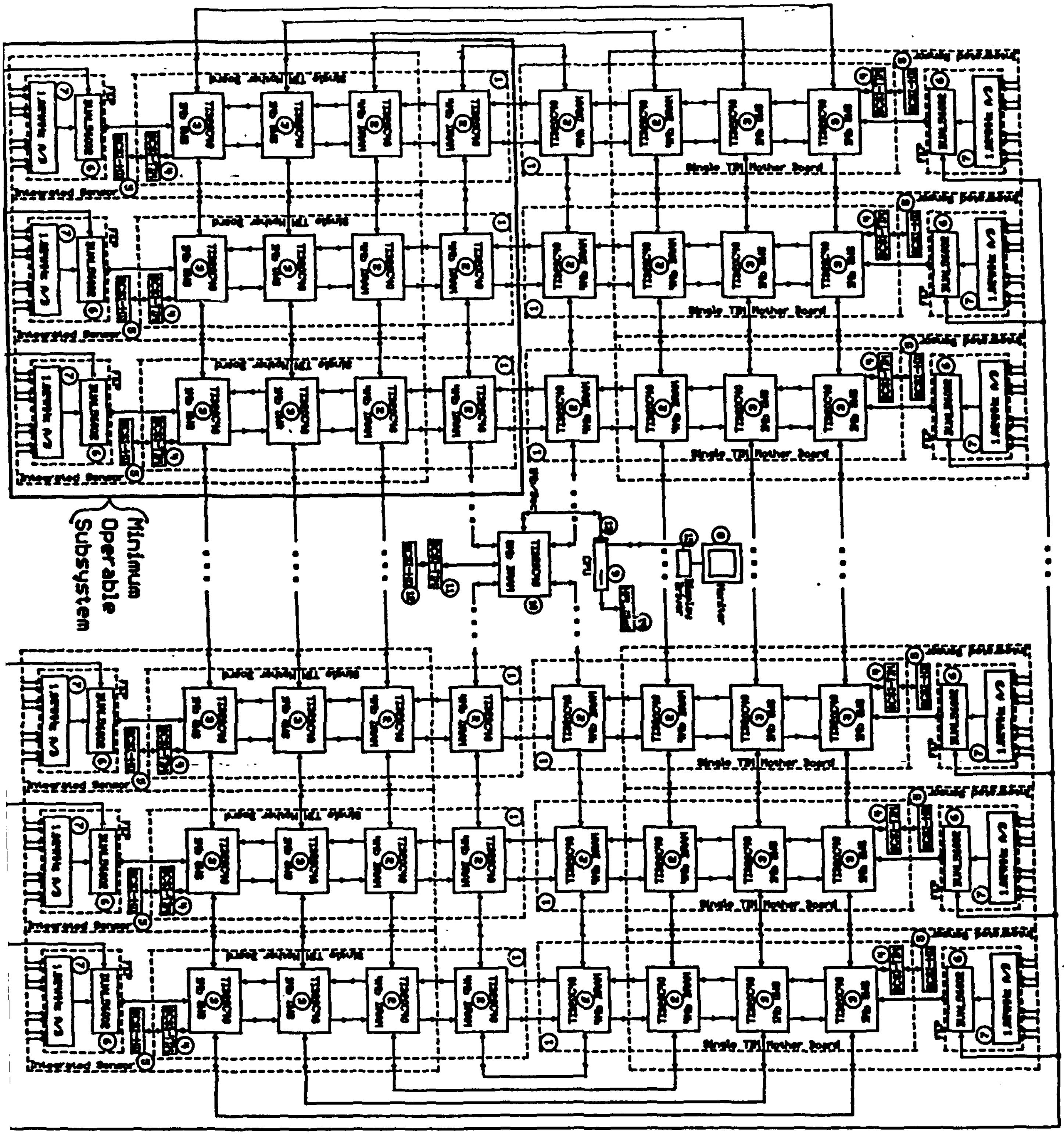

8000
7000
6000
5000
4000
3000
2000
1000
0
3000
4000
5000
6000
7000
8000
9000
10
9
8
7
6
5
4
3
2
1
AMPL: 2.00

Processing Algorithms

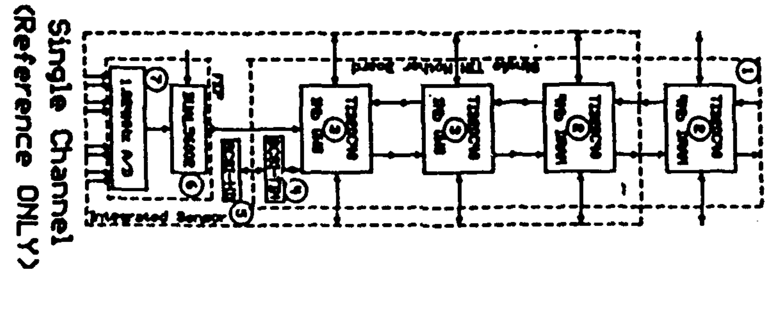


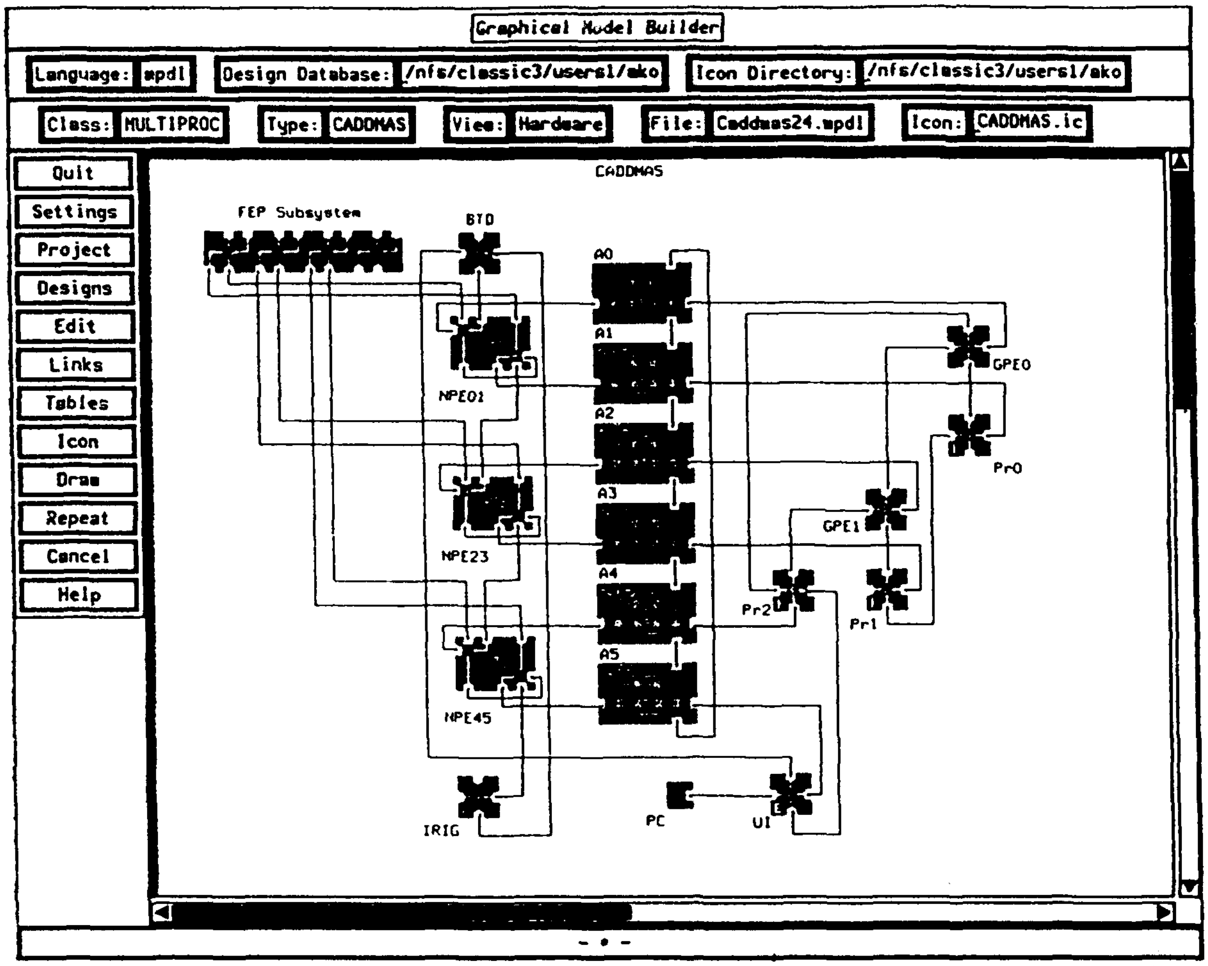
30 Ops/Pt	4 Ops/Pt	60 Ops/Pt	60 Ops/Pt	20 Ops/Pt
4 MOPS	.4 MFLOPS	6 MFLOPS	6 MFLOPS	2 MFLOPS
192 MOPS	19.2 MFLOPS	288 MFLOPS	288 MFLOPS	96 MFLOPS

Total: Approx 883 MFLOPS Sustained

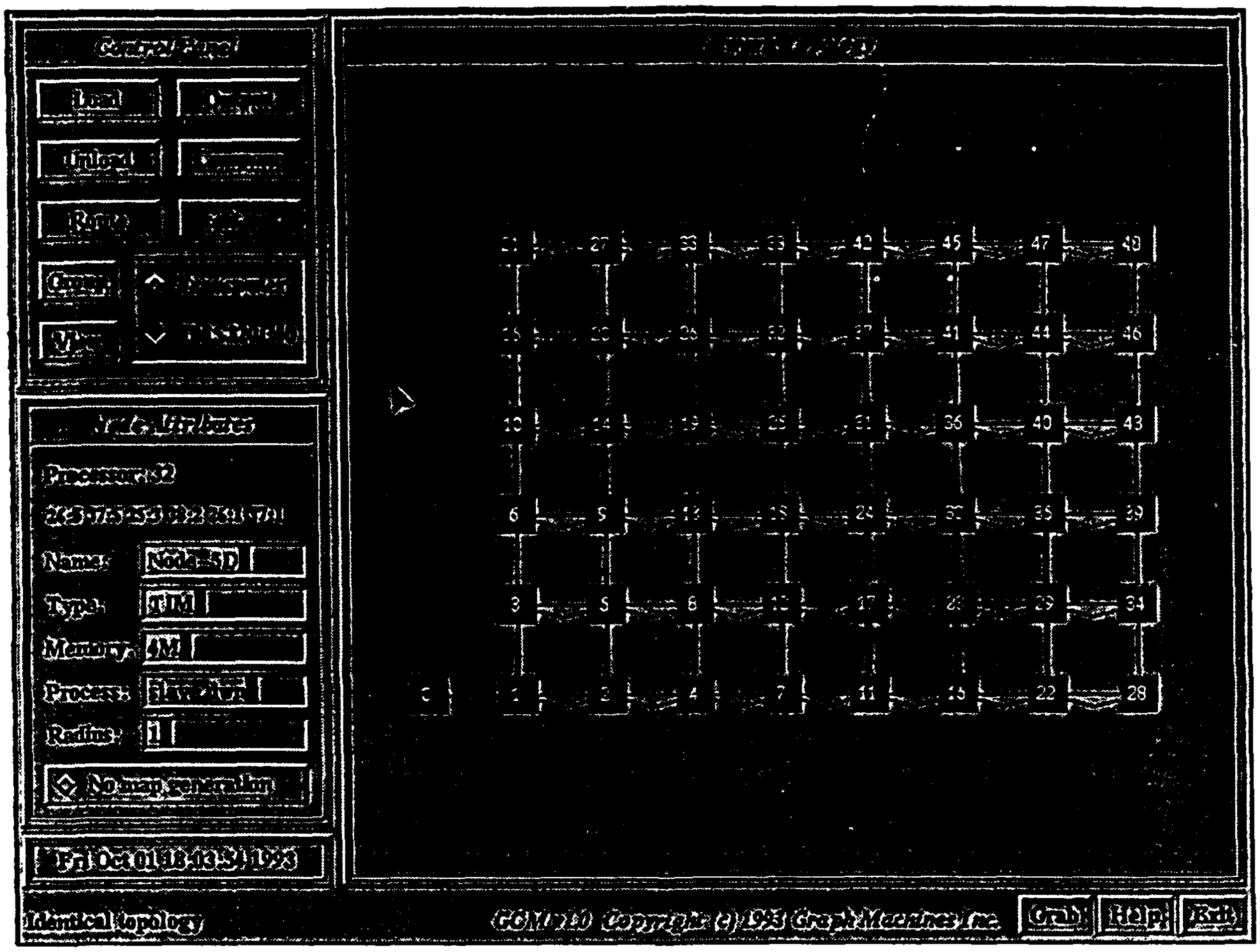


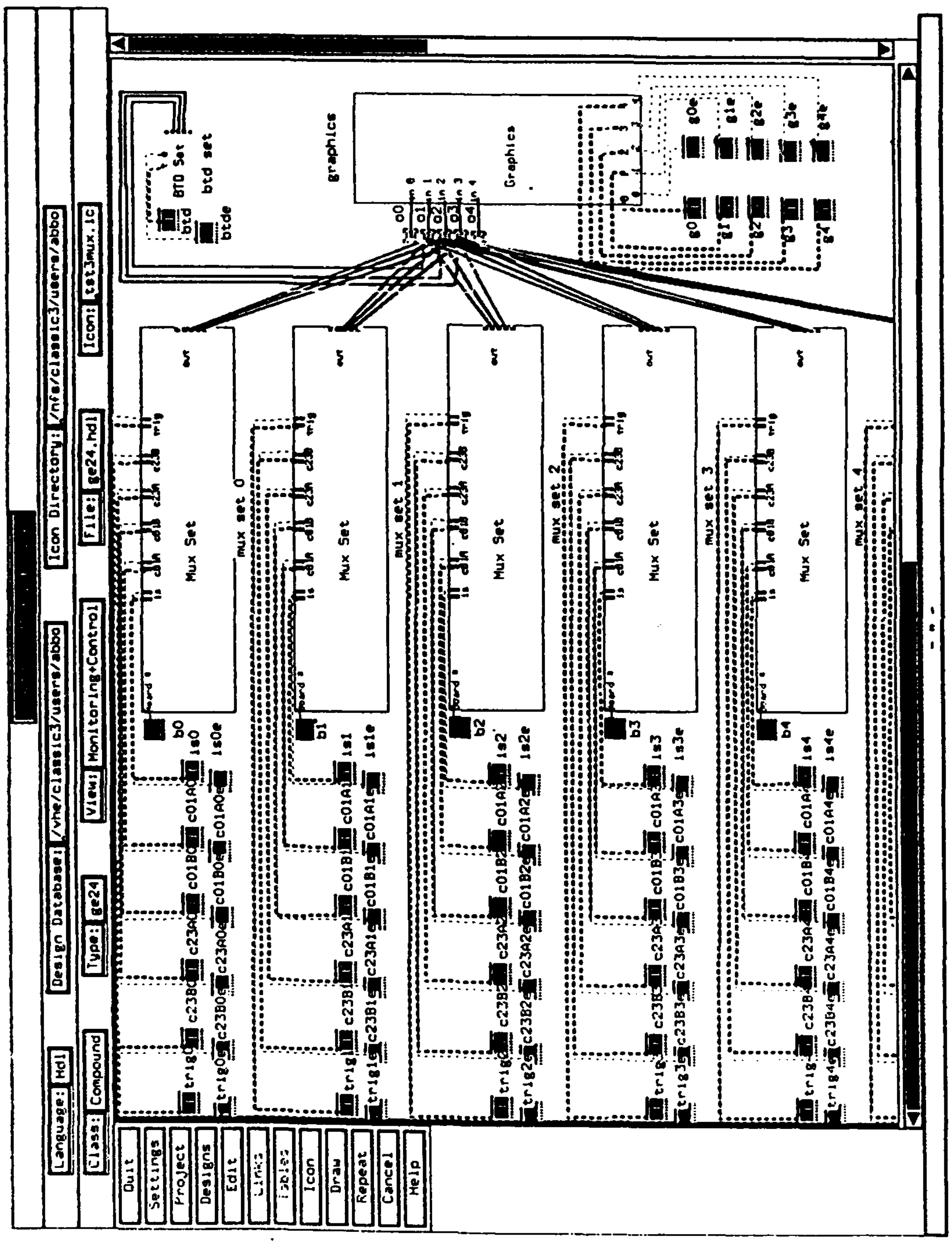
24 Channel TI-TMS320C40-Based CADDMAS
(ONLY 12 Channels Shown)

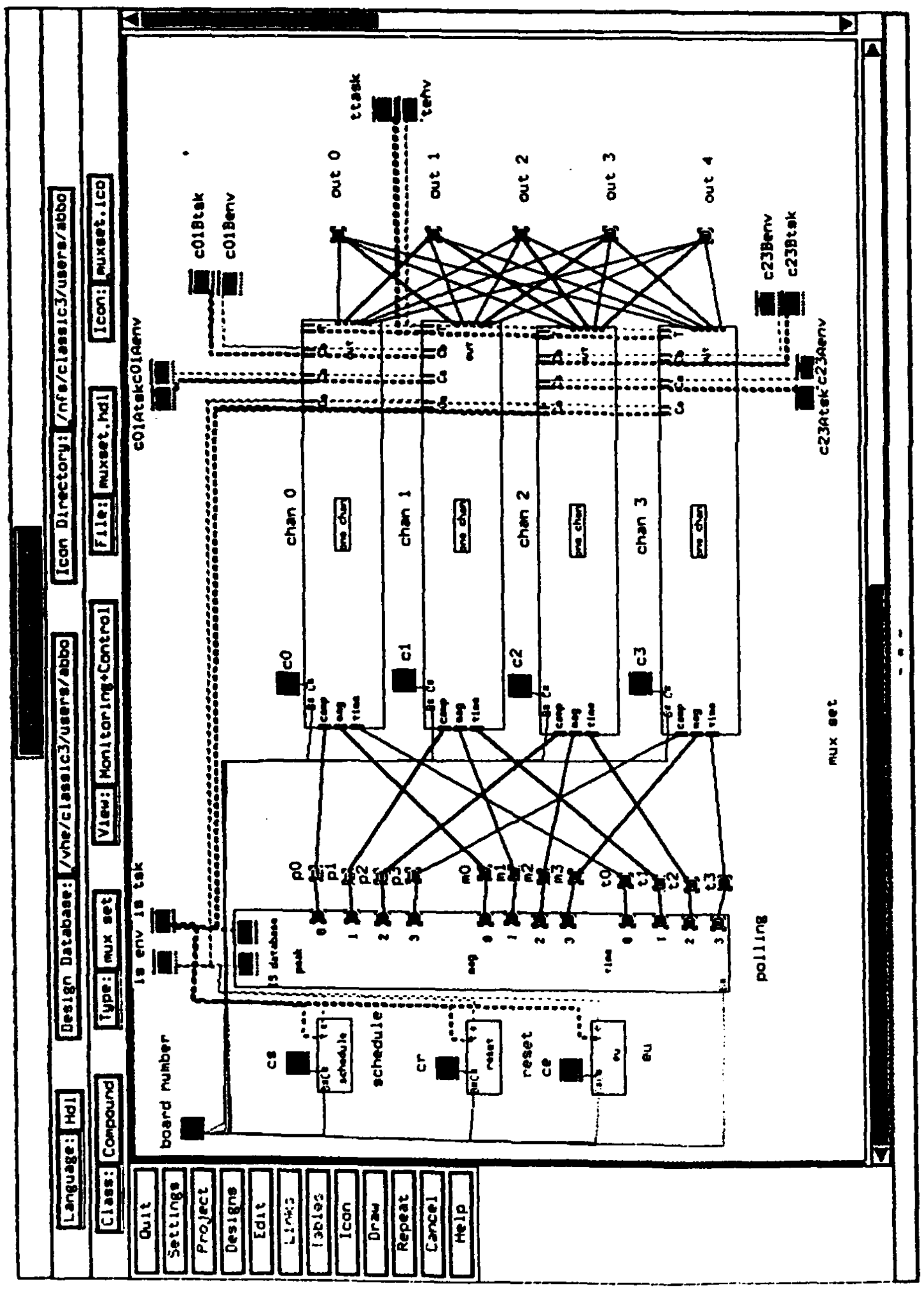


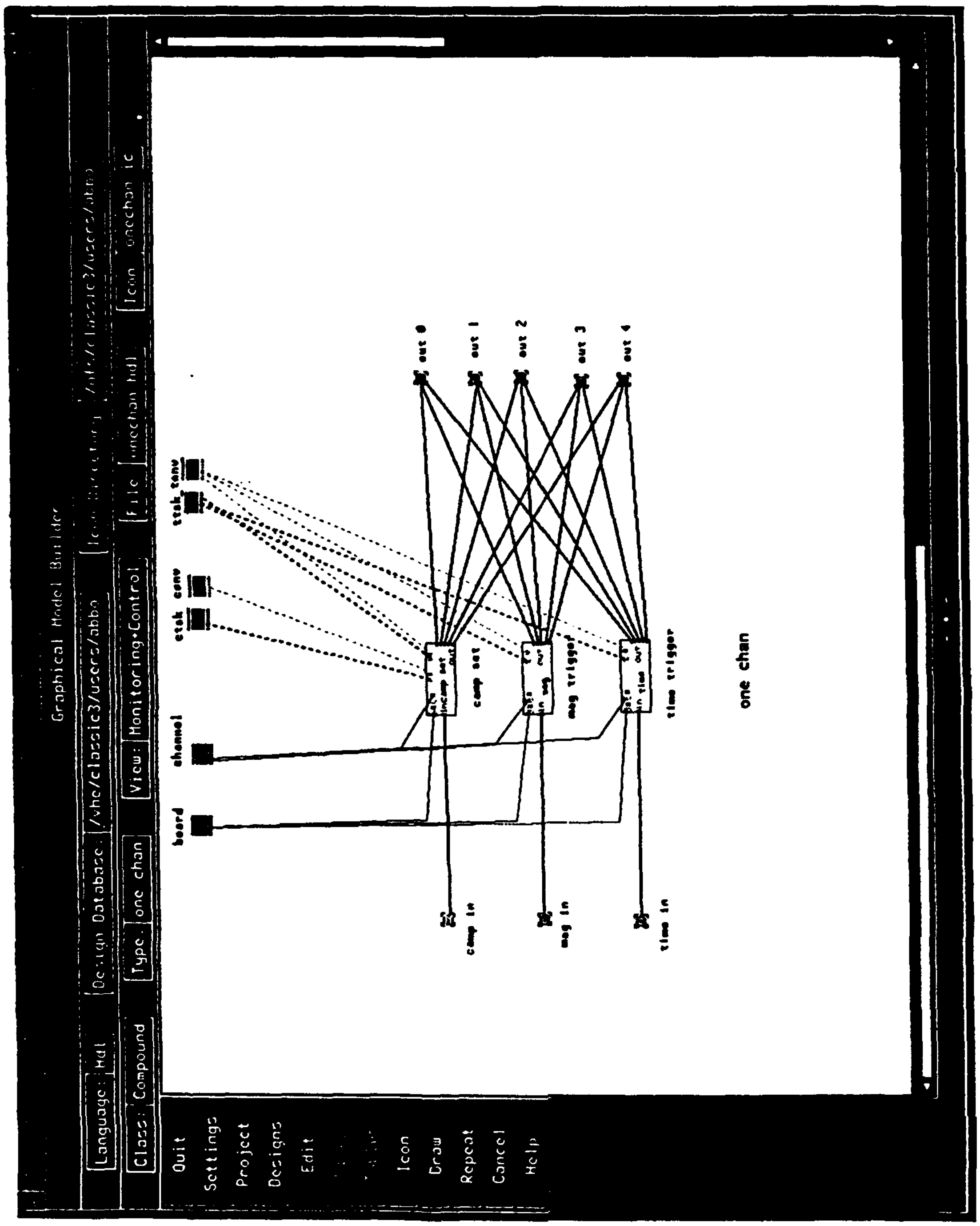

Graphical Model Builder
Language: mpdl
Design Database: /nfs/classic3/users1/mko
Icon Directory: /nfs/classic3/users1/mko
Class: MULTIPROC
Type: CADDMAS
View: Hardware
File: Caddmas24.mpdl
Icon: CADDMAS.ic
Quit
Settings
Project
Designs
Edit
Links
Tables
Icon
Draw
Repeat
Cancel
Help
CADDMAS
FEP Subsystem
BTD
A0
A1
A2
A3
A4
A5
NPE01
NPE23
NPE45
GPE0
Pr0
GPE1
Pr2
Pr1
IRIG
PC
UI

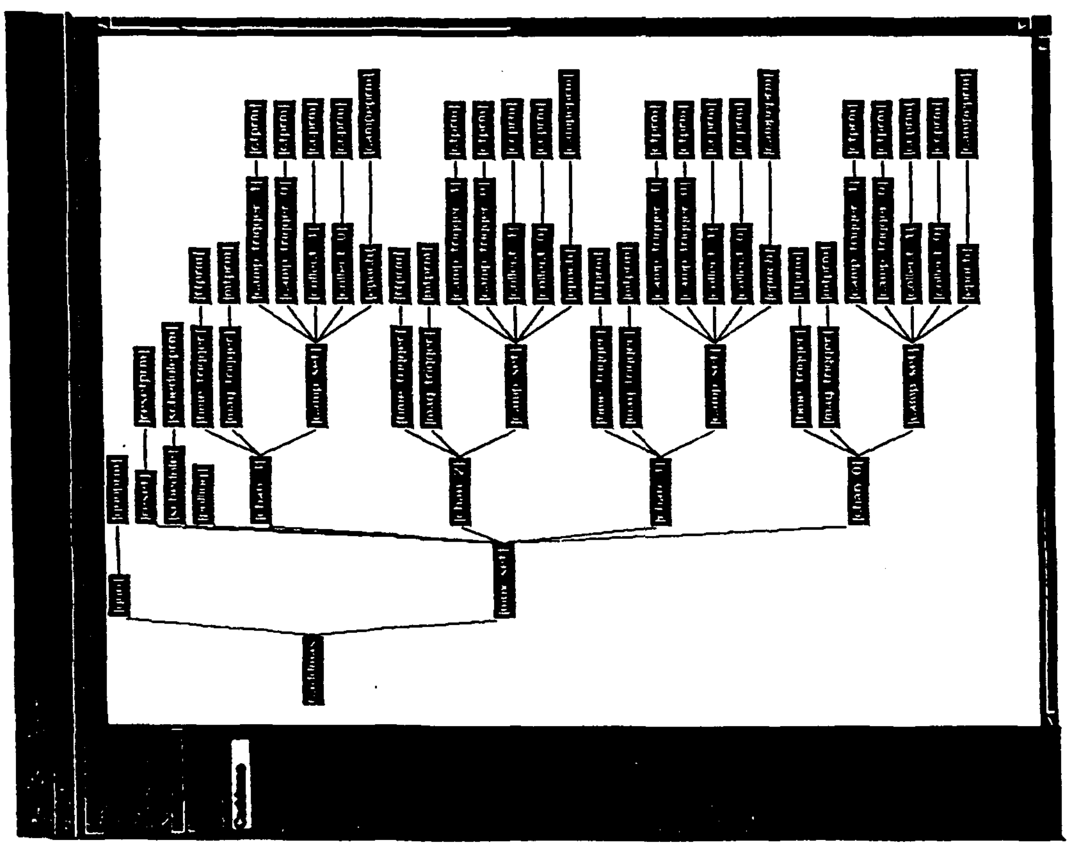
HARDWARE MODELING



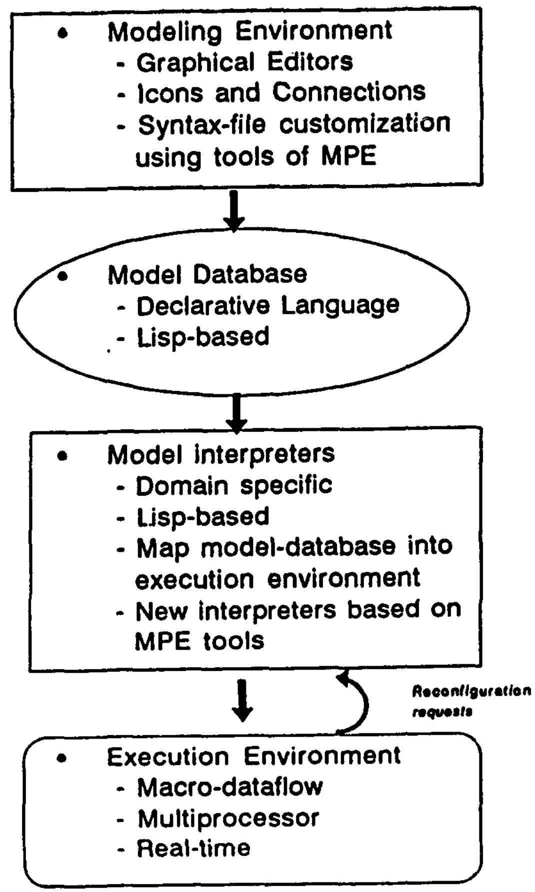

Language: Hdl
Design Database: /vhe/classic3/users/abbo
Icon Directory: /nfs/classic3/users/abbo
Class: Compound
Type: ge24
View: Monitoring+Control
File: ge24.hdl
Icon: tst3mux.ic
Quit
Settings
Project
Designs
Edit
Links
Tables
Icon
Draw
Repeat
Cancel
Help
graphics
Graphics
BTD Set
btd set
btd
btde
Mux Set
mux set 0
mux set 1
mux set 2
mux set 3
mux set 4


Language: Hdl
Design Database: /vhe/classic3/users/abbo
Icon Directory: /nfs/classic3/users/abbo
Class: Compound
Type: mux set
View: Monitoring+Control
File: muxset.hdl
Icon: muxset.ico
Quit
Settings
Project
Designs
Edit
Links
Tables
Icon
Draw
Repeat
Cancel
Help
board number
schedule
reset
polling
chan 0
chan 1
chan 2
chan 3
out 0
out 1
out 2
out 3
out 4
ttask
tenv
c01Btsk
c01Benv
c23Benv
c23Btsk
mux set

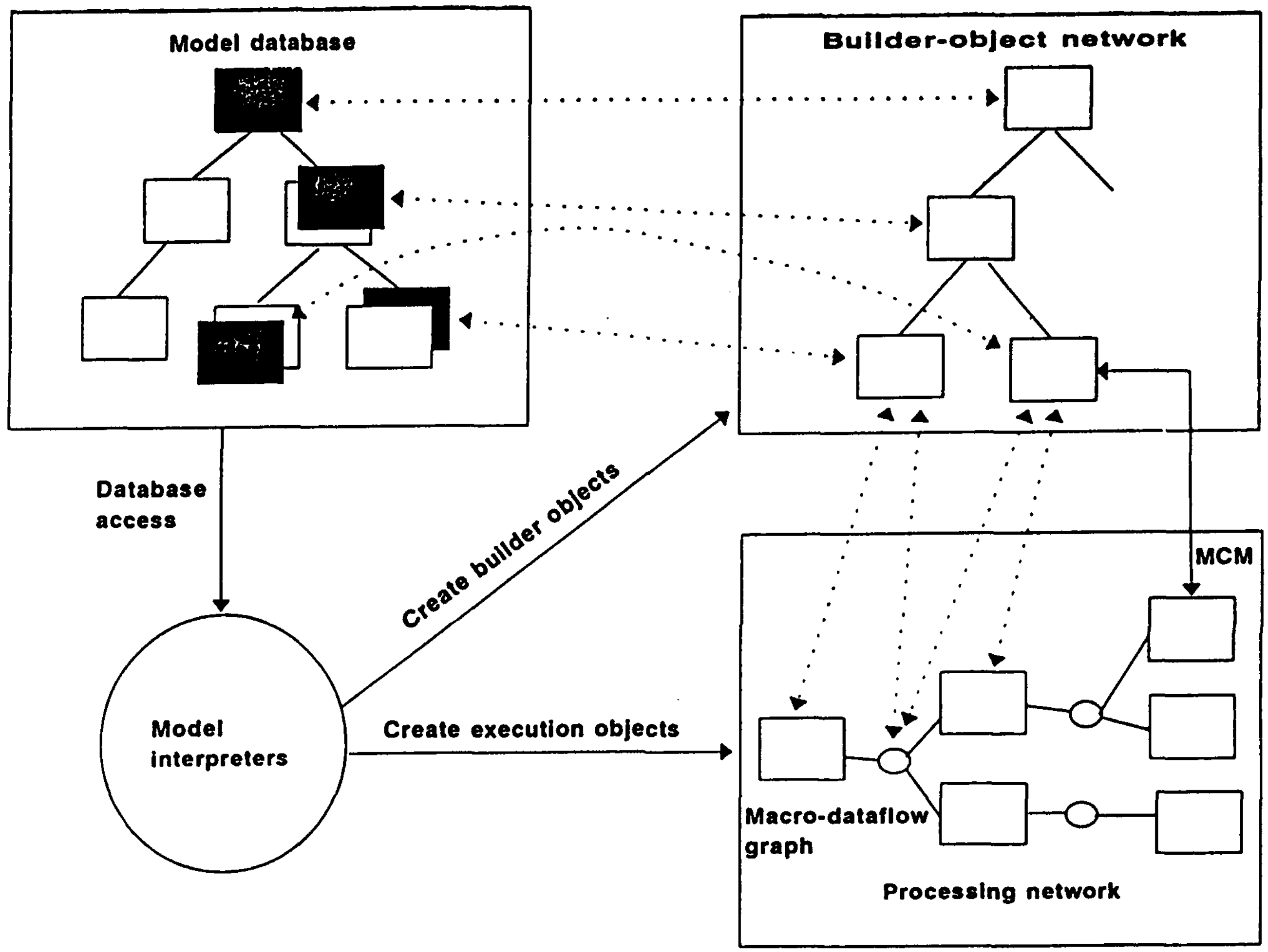

Graphical Model Builder
Language: Hdl
Design Database: /vhc/classic3/users/abba
Class: Compound
Type: one chan
View: Monitoring•Control
File
Icon
Quit
Settings
Project
Designs
Edit
Icon
Draw
Repeat
Cancel
Help
board
channel
comp in
mag in
time in
comp set
mag trigger
time trigger
out 0
out 1
out 2
out 3
out 4
one chan



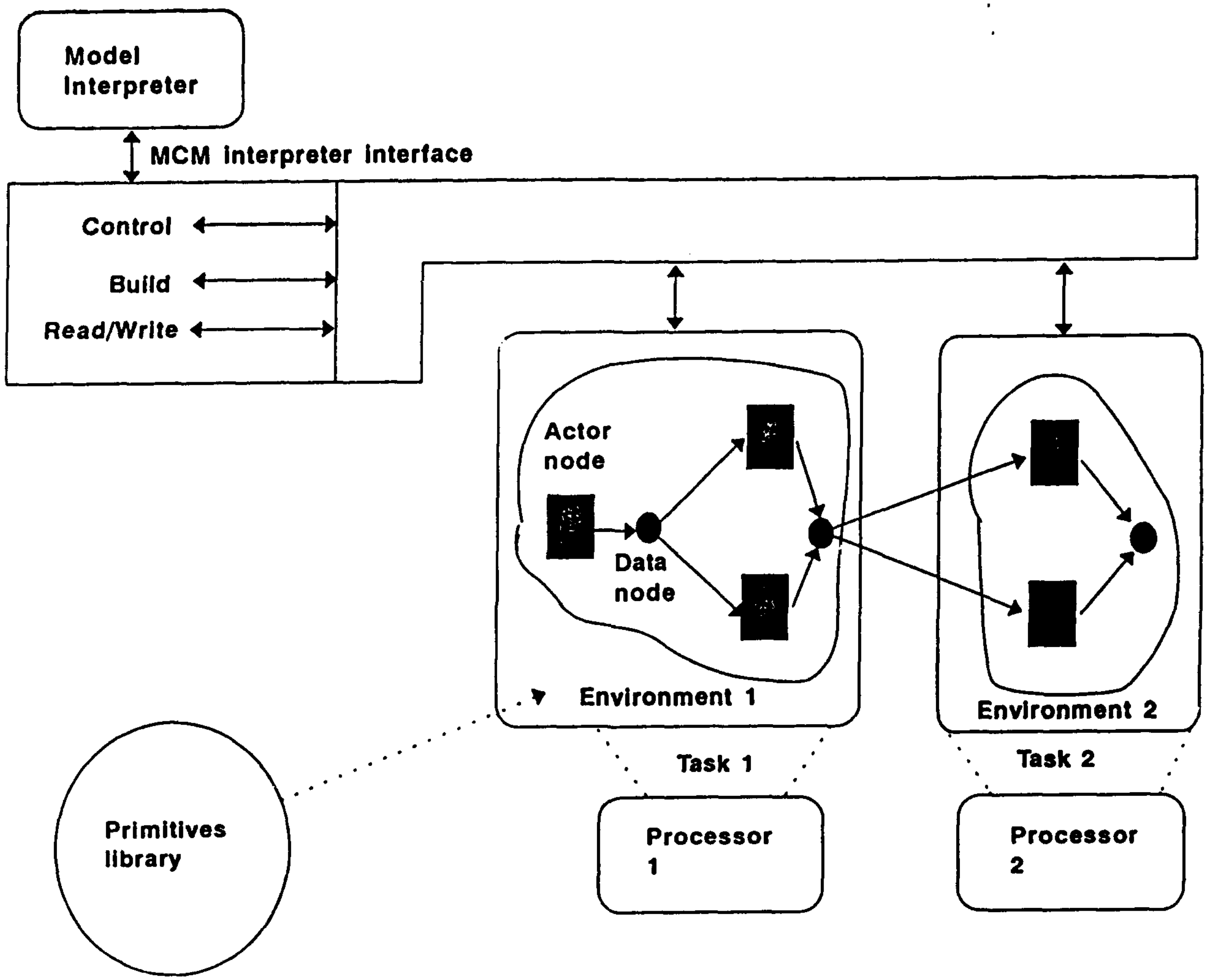
Multigraph Architecture



Model interpretation

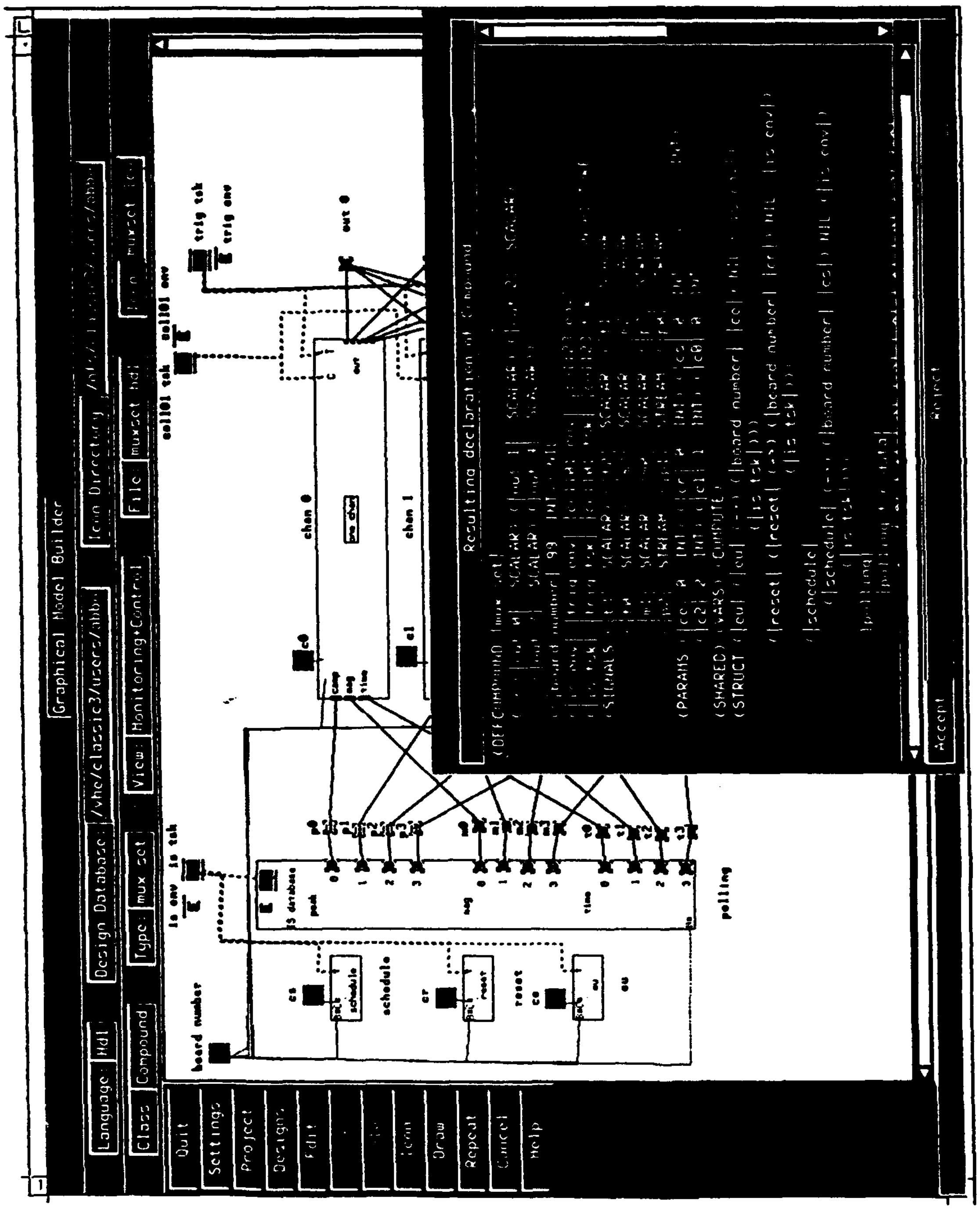


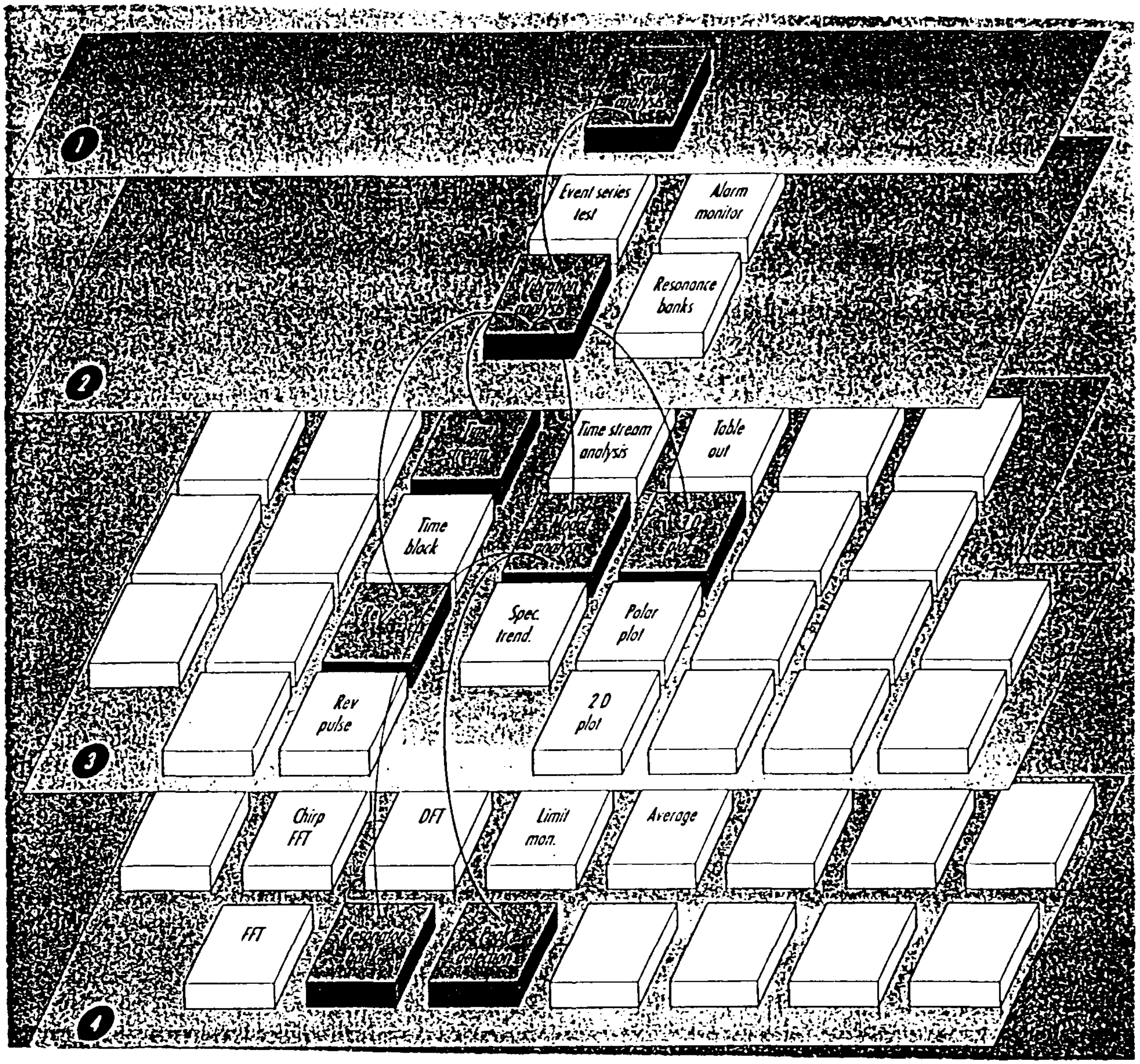
Execution Environment



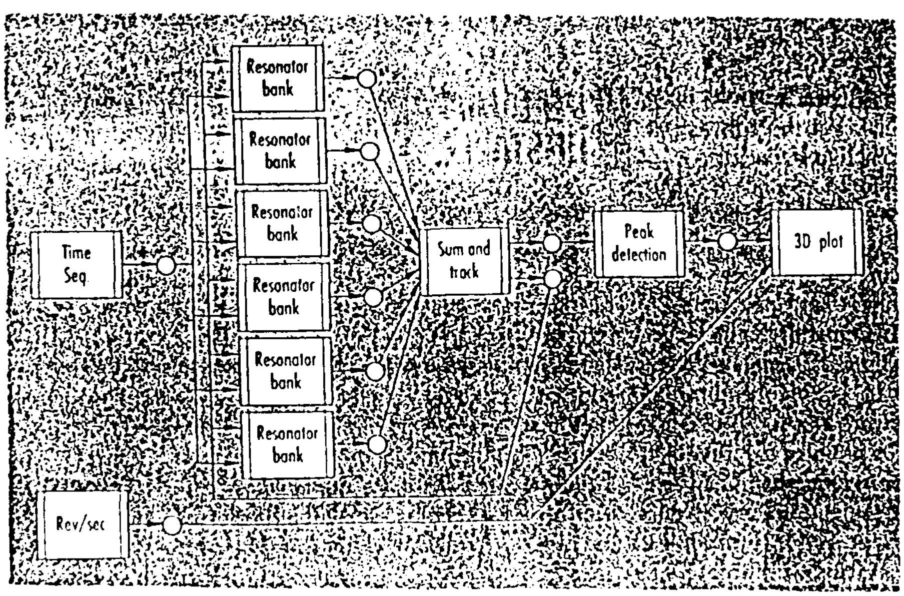
```
(defprimitive <name>
        (interface (<input-signals> -> <output-signals>)
                   ( <specification-list>)
                   ( <dynamic-control-parameters>)
        )
(body              (<primitive-name>) (<discipline>) (<environment>))


        (defcpu <name>
                (linkpoints (<linkpoint-list>))
                (specification (<specifcation-list>)))

        (defmpr <name>
                (parts (<part-list>))
                (connections (<connection-list>)))
```

```
(defcompound <name>
        (interface    (<input-signals> -> <output-signals>)

                      (<specification-list>)
                      (<dynamic-control-parameters>)
                      (<environment>))
        (connections  (<signals>) (<specifications>))
        (linkpoints   (<list of link-point specifications>))
        (structure    (<list of units>))
                      (<selection rules>)
(body   (<S-expressions>)))
```


Graphical Model Builder
Language: Hdl
Design Database: /vhe/classic3/users/abbe
Icon Directory
Class: Compound
Type: mux set
View: Monitoring+Control
File: muxset.hdl
Quit
Settings
Project
Draw
Repeat
Cancel
Help
Resulting declaration of Compound
Accept
Reject



1
Event series test
Alarm monitor
Resonance banks
2
Time stream analysis
Table out
Time block
Spec. trend.
Polar plot
Rev pulse
2 D plot
3
Chirp FFT
DFT
Limit mon.
Average
FFT
4



Resonator bank
Resonator bank
Resonator bank
Resonator bank
Resonator bank
Resonator bank
Time Seq.
Sum and track
Peak detection
3D plot
Rev/sec

Creating a New CADDMAS

- Model additional basic processing elements (provide computation subroutines if needed)
- Model new hardware elements
- Synthesize architecture from:
  - Model Database
  - Specifications
- Wire the hardware
  - Verify against models
- Synthesize and run the system

A New Domain

- Define Modeling Paradigm / Concepts
- Build Editors
- Build Interpreters
- Produce Basic Computation Library

Conclusion

- Models aid in complexity management
- Domain specific models are helpful
- Hardware system design is integrated with software
- Performance is not compromised

PREMOS

Programming Environment for Model-based Program Synthesis

Hubertus Franke

IBM T.J.Watson Research Center
Yorktown Heights, NY 10598

Model-Based Programming Environments (MPE)

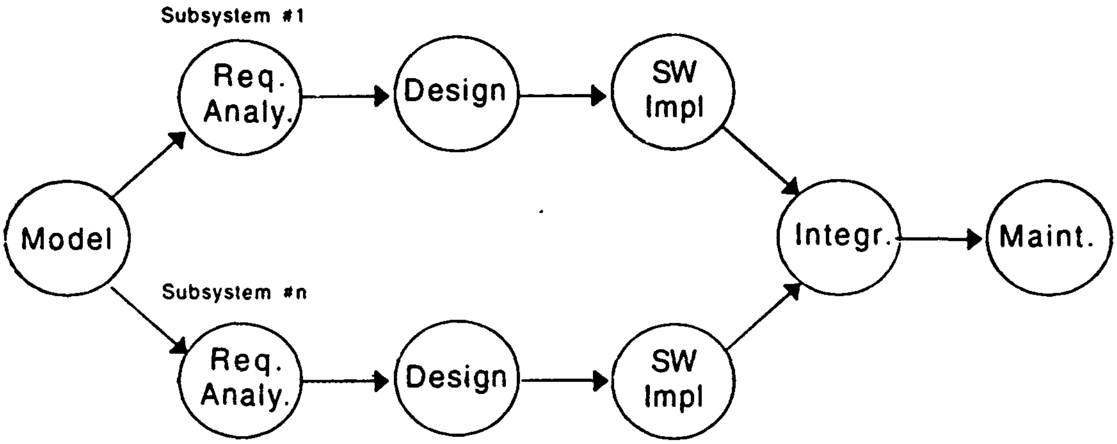
What is the need for MPE ?

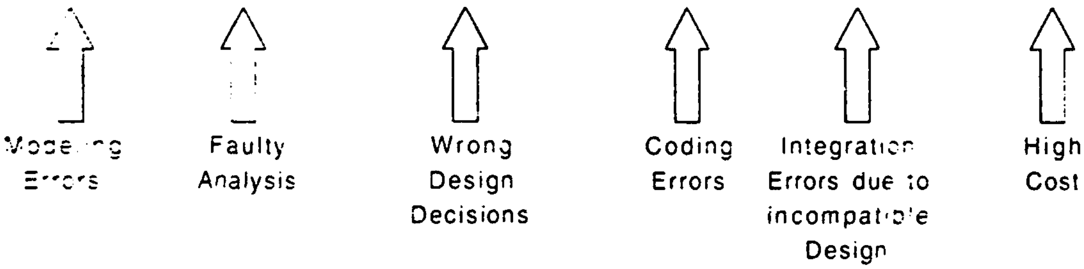
- Non-software engineers want to develop complex software
- Minimal use of traditional programming language
- Users are interested in domain engineering not in software engineering

Requirements for a MPE :

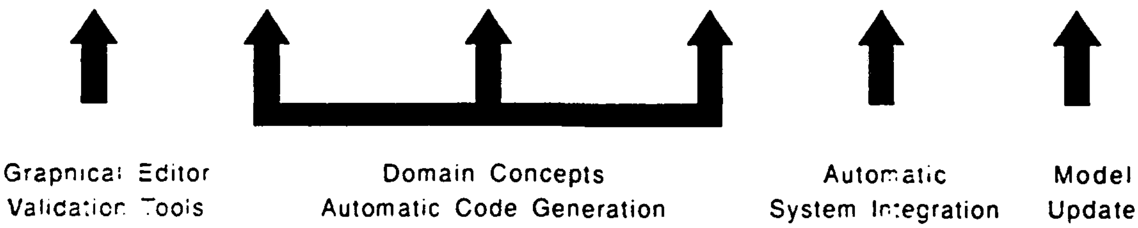
- Capture precise representation of system to be build (module topology, interface specification, hierarchy, data flow, architecture, ...)
- Concepts close to the domain
- No hassle with system generation and integration
- Limited programming required
- Robust but flexible to changing requirements

Conventional Software Engineering Approach





Inherent Error Sources

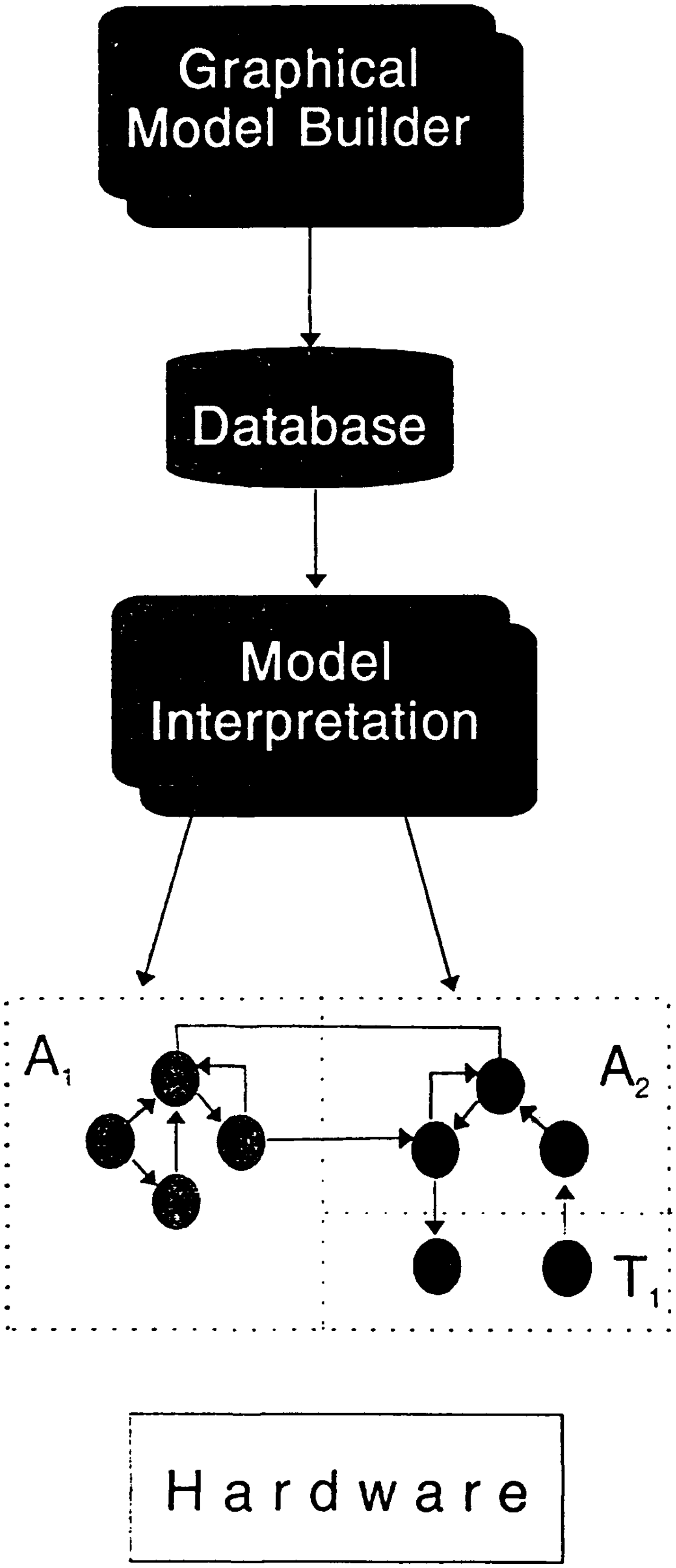


MPE to the Rescue

Increase in :

- **Productivity**
- **Maintenaince**
- **Reusuability**
- **Documentation**
- **Users**

Key Concepts



- iconic
- attributes
- multiple aspects

- multi user
- object-oriented

- multiple aspects
- automatic program generation
- automatic system integration
- incremental

- virtual concurrent program

- run-time objects

- target system (s)

- parallel
- hybrid

Tools' Perspective

Characteristics:

- Technology = Concepts + Tools
- Tools are very expensive to build (200 KL)
- Very domain dependent
  centered around the Modeling Paradigm
- Modeling Paradigm is an ever changing Entity

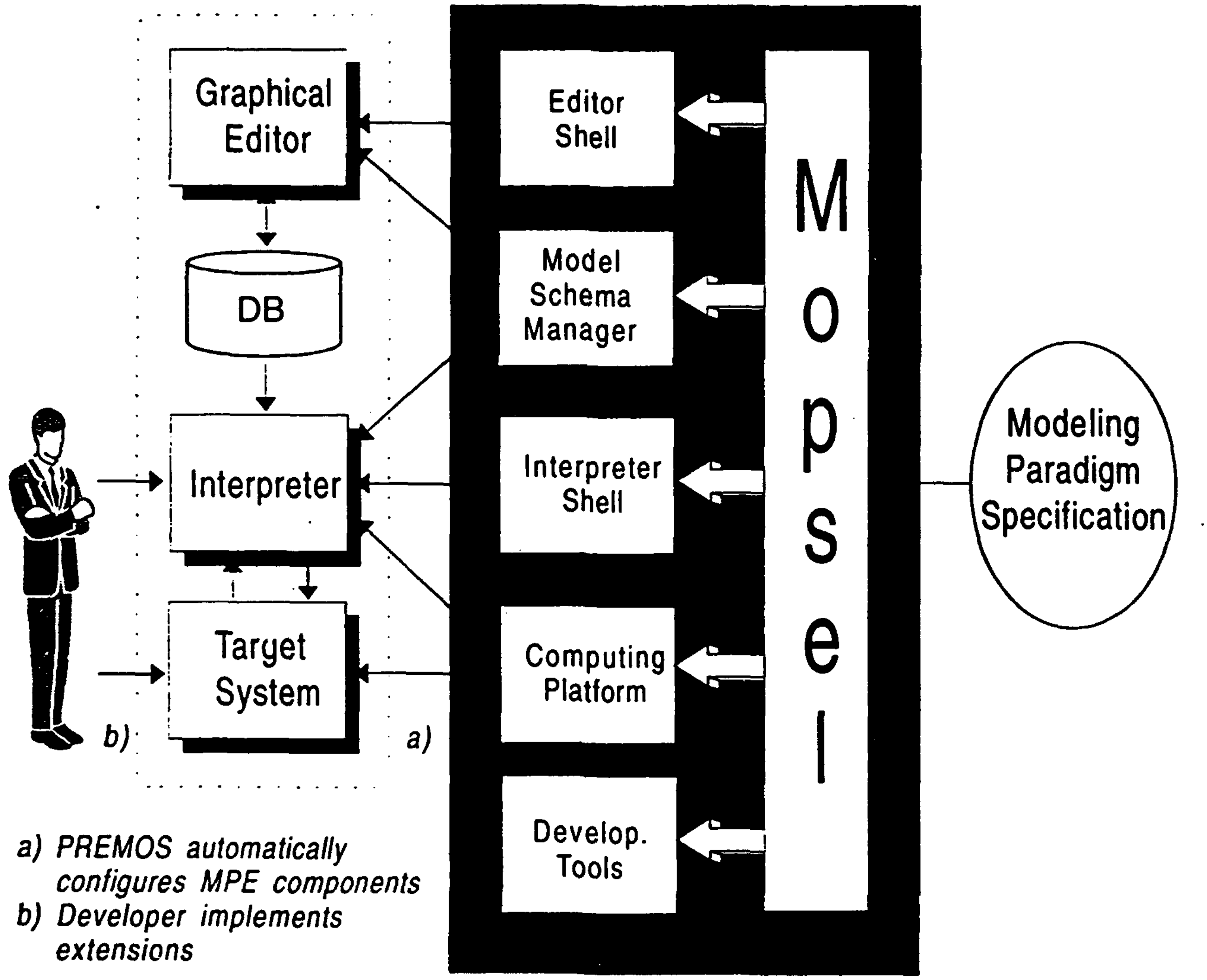
→ *Tool Development undergoes similar Software Engineering Cycle as Systems targeted by Tools*

Approach:

- Take a "model-based" Approach

PREMOS

- formally specify modeling paradigm
- compile specification and automate MPE system generation

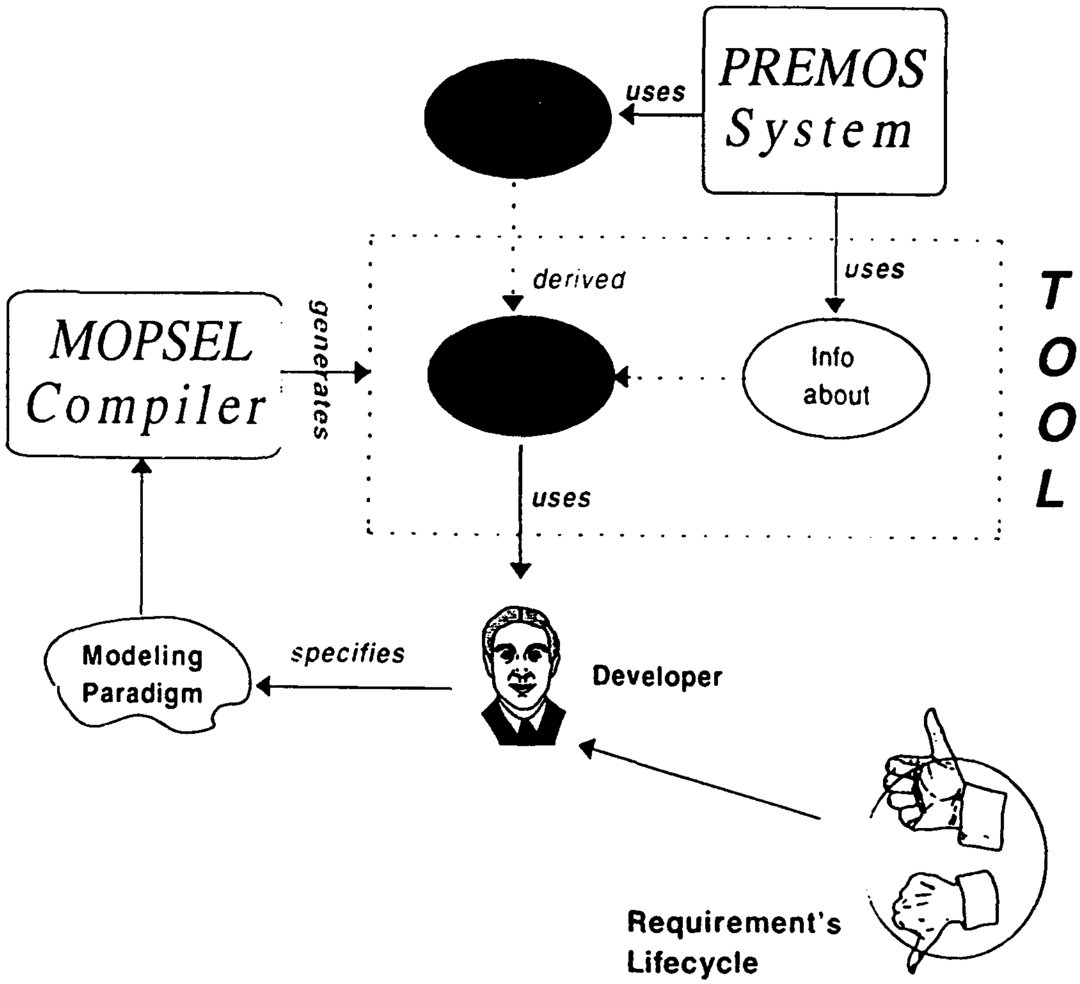


a) PREMOS automatically configures MPE components
b) Developer implements extensions

Standardization vs. Specialization

Implement Tools based on generic Concepts and customize them based on Modeling Paradigm

C++ - based Implementation



Modeling Paradigm and various Key Components

Model Concepts

- Iconic Editor
- Attribute Editor
- Hierarchies + Cross References
- Database Interface
- Schema Compiler

Object Oriented Database

Interpretation Language

- Model Traversal
- Reference Resolution
- Interpretation attributes

Runtime Objects

- strong virtual machine layer
- various concept, e.g.
- light weight threads
- actor models

Intelligent Process Control Domain

Independent System Aspects

Equipment Aspect

Process Aspect

Activity Aspect

Dependent Aspects

- Monitoring and Control
  Signals, Events, Alarms, Primitives, Compounds

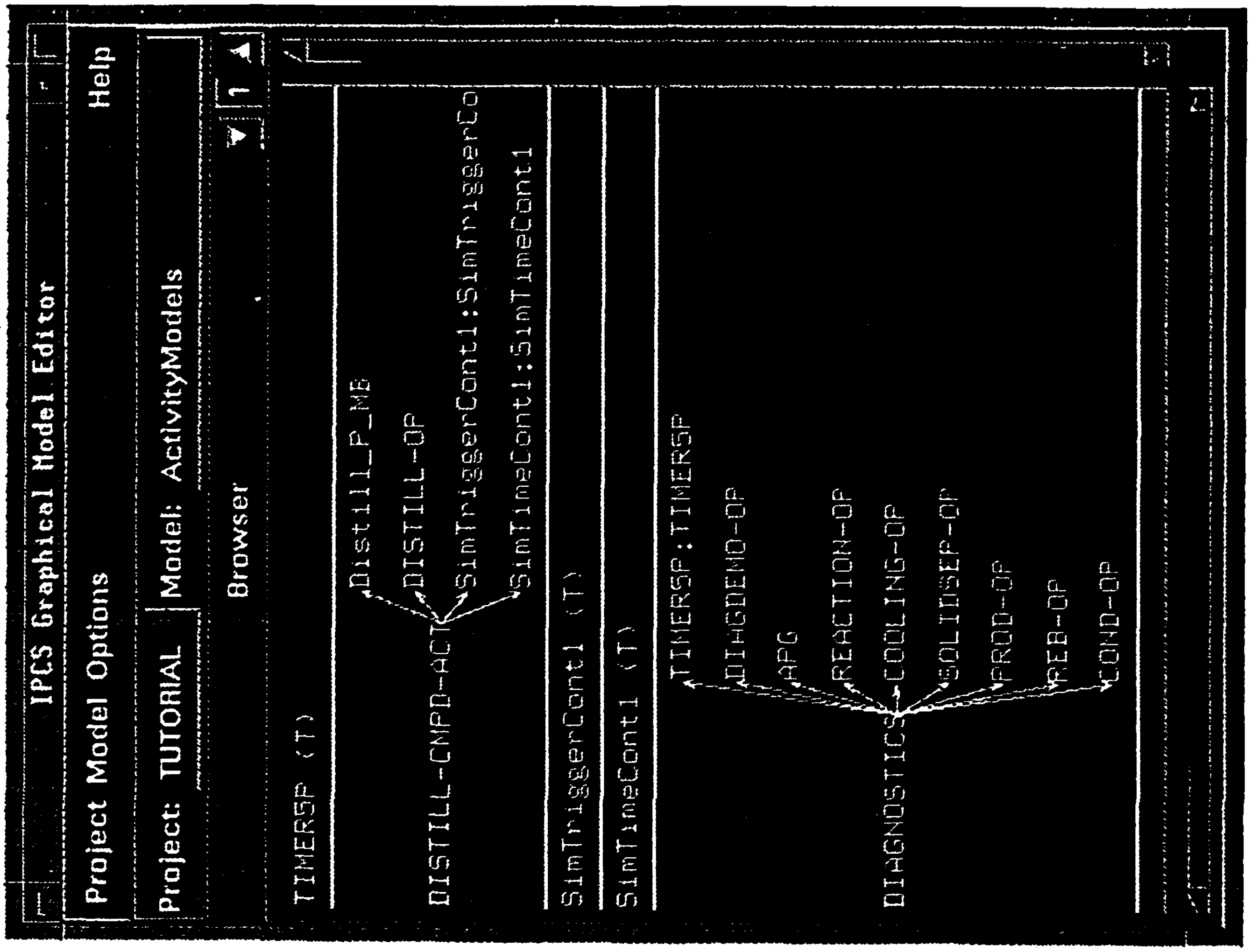
- Finite State Machine
  State, StateMachines

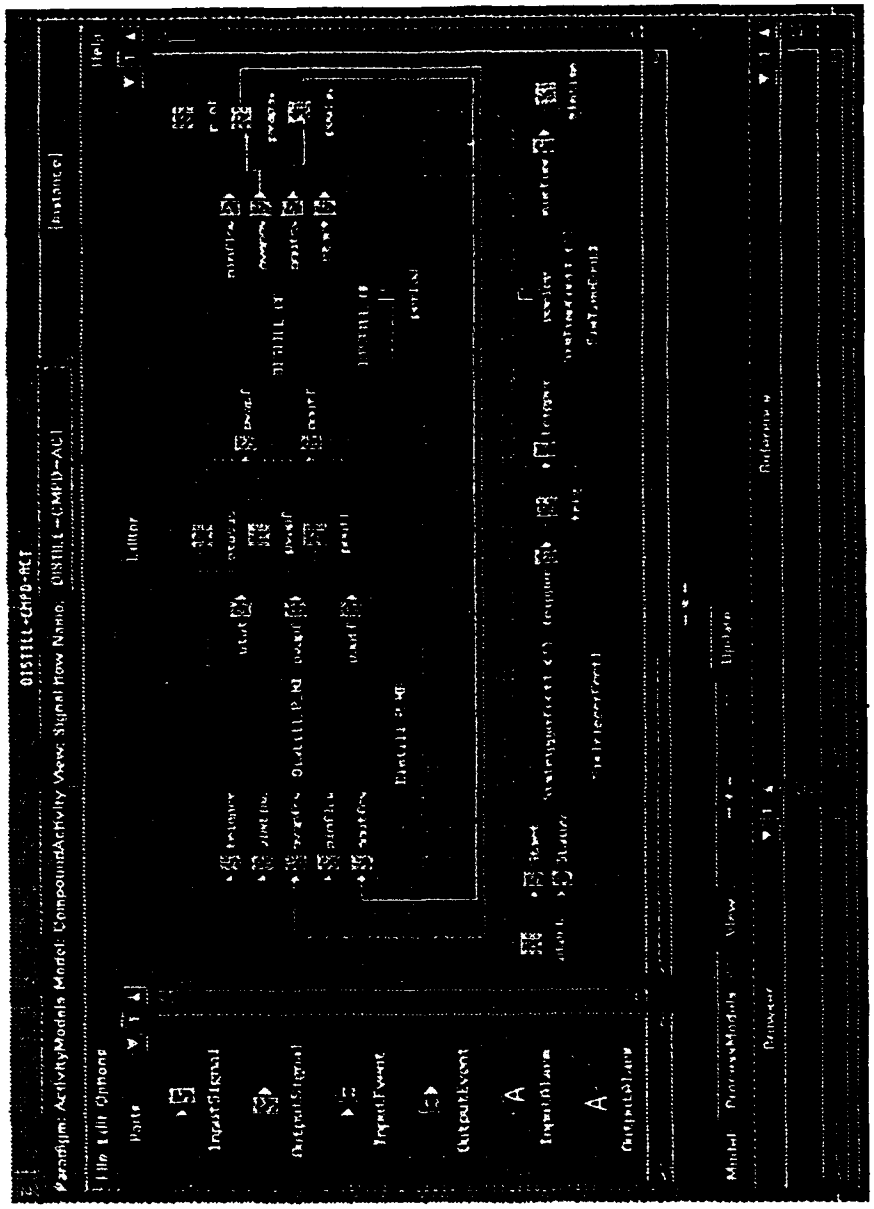
- Operator Interface
  Panels, Button, Graphs

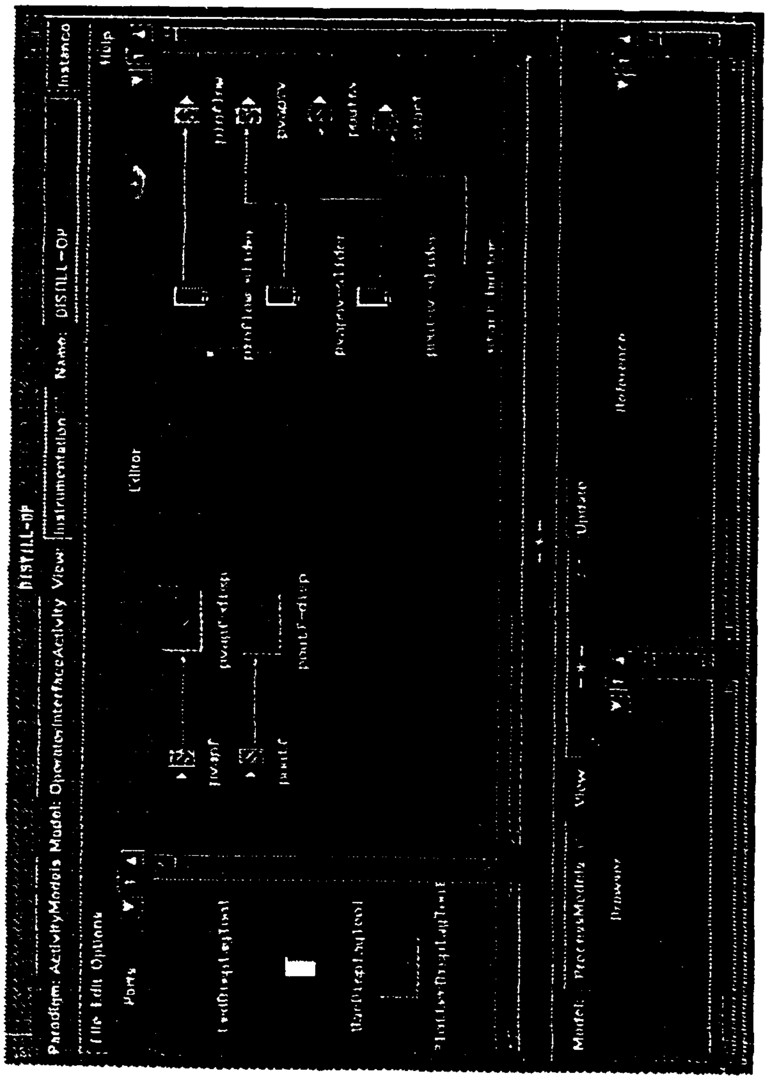
- Diagnostics
  FaultMode, Propagation

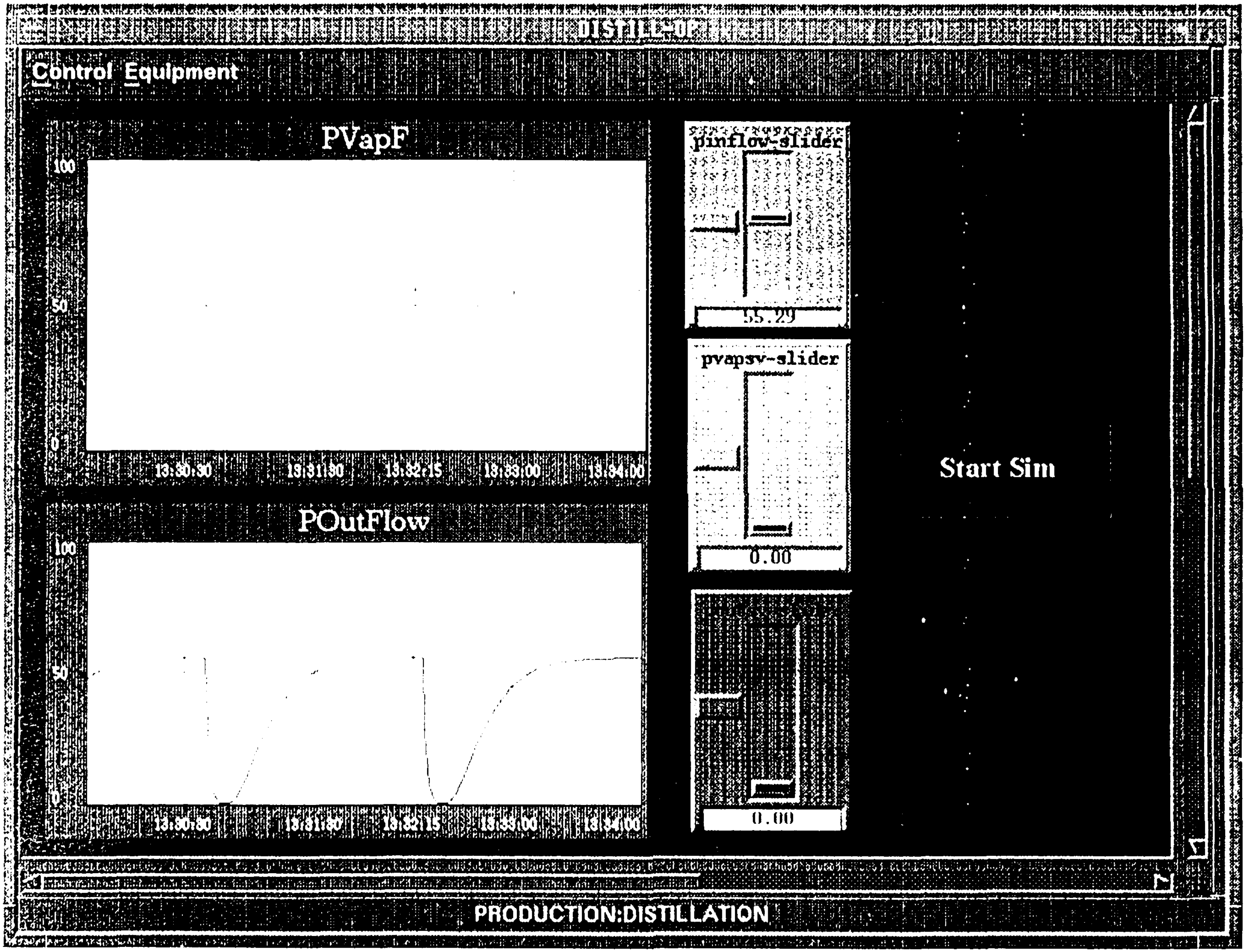
```
#deficondir "icons" ;

[illegible]
#include "hdl.cdf"


#define HDLModels # cHDL
Icons:
        InputSignal = cISignal : "isignal.icon"
                                { Type # cSignalType : menu "Select signal type:"
                                       { "Stream" #cStreamSignal;
                                         "Scalar" #cScalarSignal;
                                       };
                                };
        OutputSignal = cOSignal : "osignal.icon"
                                { Type # cSignalType : menu "Select signal type:"
                                       { "Stream" #cStreamSignal;
                                         "Scalar" #cScalarSignal;
                                       };
                                };
        LocalSignal = cLSignal : "lsignal.icon"
                                { Type # cSignalType : menu "Select signal type:"
                                       { "Stream" #cStreamSignal;
                                         "Scalar" #cScalarSignal;
                                       };
                                };
        InputParameter = cIParameter : "iparam.icon"
                                { Type # cParamType : menu "Select parameter type:"
                                       { "Value"     #cValueParameter;
                                         "Reference" #cReferenceParameter;
                                       };
                                };
        LocalParameter = cLParameter : "lparam.icon"
                                { Type # cParamType : menu "Select parameter type:"
                                       { "Value"     #cValueParameter;
                                         "Reference" #cReferenceParameter;
                                       };
                                };
```

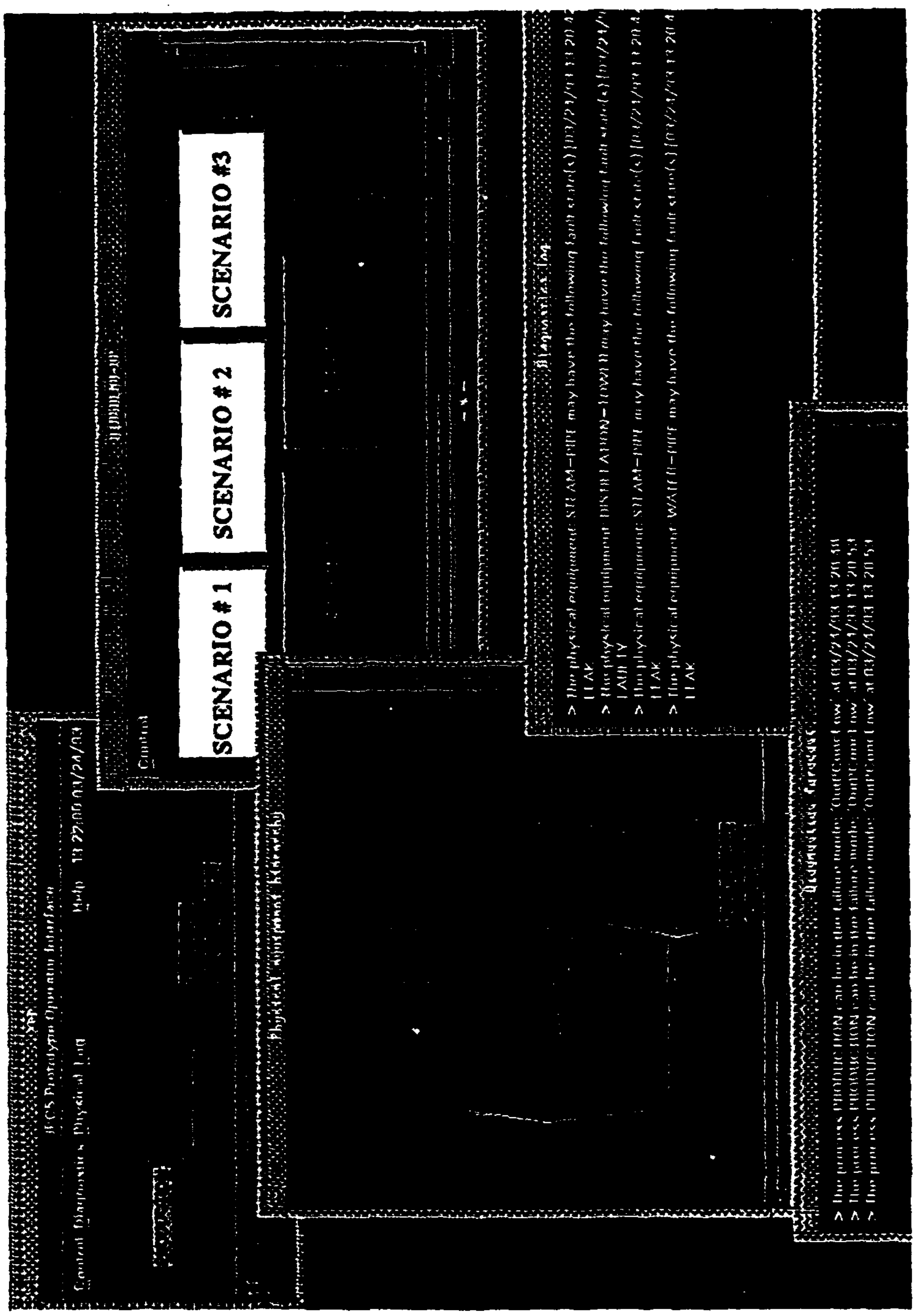
```
models:
  Primitive # cHDLPrimitive {
    views:
      'Signal flow' = cSignalFlow {
      icon rect {, left   : InputSignals;
                   right  : OutputSignals;
                   top    : InputParameters;
                 };
      font #2;
      color foreground;
      attributes {
              Script = cScript : page "Primitive script:" ( 8 40 ) ""
                                      { 0 0 1 2 };
              Control  = cControl : menu "TriggerMode:"
                                            { "IfAll" #cIfany; "IfAny" #cIfall; }
                                         { 1 0 1 1 };
      }
      parts {
        InputSignals     = cInputSignal     : InputSignal     link;
        OutputSignals    # cOutputSignal    : OutputSignal    link;
        InputParameters = cInputParameter : InputParameter link;
    } ; }
  Compound = cHDLCompound {
    views:
      'Signal flow' = cSignalFlow {
      icon rect { left   : InputSignals;
                  right  : OutputSignals;
                  top    : InputParameters;
                 };
      font #2;
      color foreground;
      conns {
              DataflowConn = cDataflowConn { 1 solid line arrow } :
              { InputSignals -> Blocks InputSignals single }
              { LocalSignals -> Blocks InputSignals single }
              { Blocks OutputSignals single -> LocalSignals }
              { Blocks OutputSignals single -> OutputSignals };
              ParameterConn = cParameterConn { 1 dash1_1 line arrow } :
              { InputParameters -> Blocks InputParameters }
              { LocalParameters -> Blocks InputParameters };
      }
      parts {
        InputSignals     = cInputSignal     : InputSignal     link;
        OutputSignals    = cOutputSignal    : OutputSignal    link;
        LocalSignals     = cLocalSignal     : LocalSignal;
        InputParameters = cInputParameter : InputParameter link;
        LocalParameters = cLocalParameter : LocalParameter link;
        SignalRefs       = cSignalRef ->
                           HDLModels : Compound : 'Signal flow' : LocalSignals };
        Blocks           = cPBlock          : Primitive;
        Blocks           = cCBlock          : Compound;
    } } }
end
```


IPCS Graphical Model Editor
Project Model Options
Help
Project: TUTORIAL
Model: ActivityModels
Browser
TIMERSP (T)
DISTILL-CMPD-ACT
Distill_P_MB
DISTILL-OP
SimTriggerCont1:SimTriggerCo
SimTimeCont1:SimTimeCont1
SimTriggerCont1 (T)
SimTimeCont1 (T)
DIAGNOSTICS
TIMERSP:TIMERSP
DIAGDEMO-OP
APG
REACTION-OP
COOLING-OP
SOLIDSEP-OP
PROD-OP
REB-OP
COND-OP



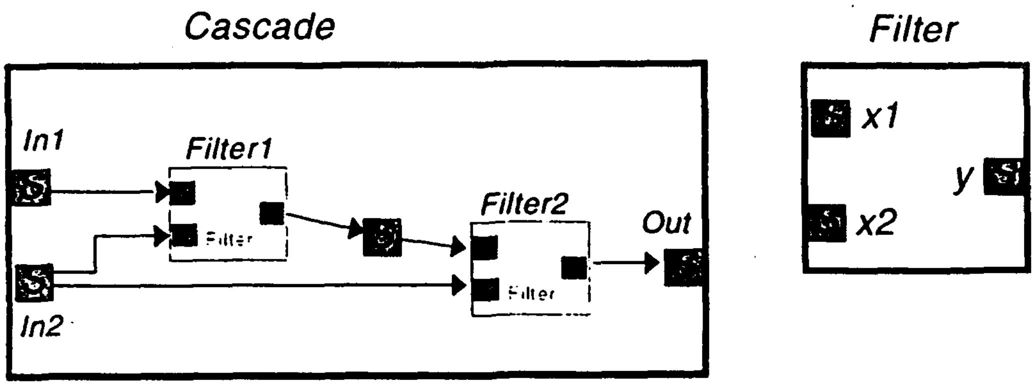

DISTILL-OP
Paradigm: ActivityModels Model: OperatorInterfaceActivity View: Instrumentation Name: DISTILL-OP
Instance
File Edit Options
Help
Ports
Editor
Model: ProcessModels View
Update
Browser
Reference


Control Equipment
PVapF
100
50
0
13:30:30
13:31:30
13:32:15
13:33:00
13:34:00
POutFlow
100
50
0
13:30:30
13:31:30
13:32:15
13:33:00
13:34:00
pinflow-slider
55.29
pvapsv-slider
0.00
0.00
Start Sim
PRODUCTION:DISTILLATION

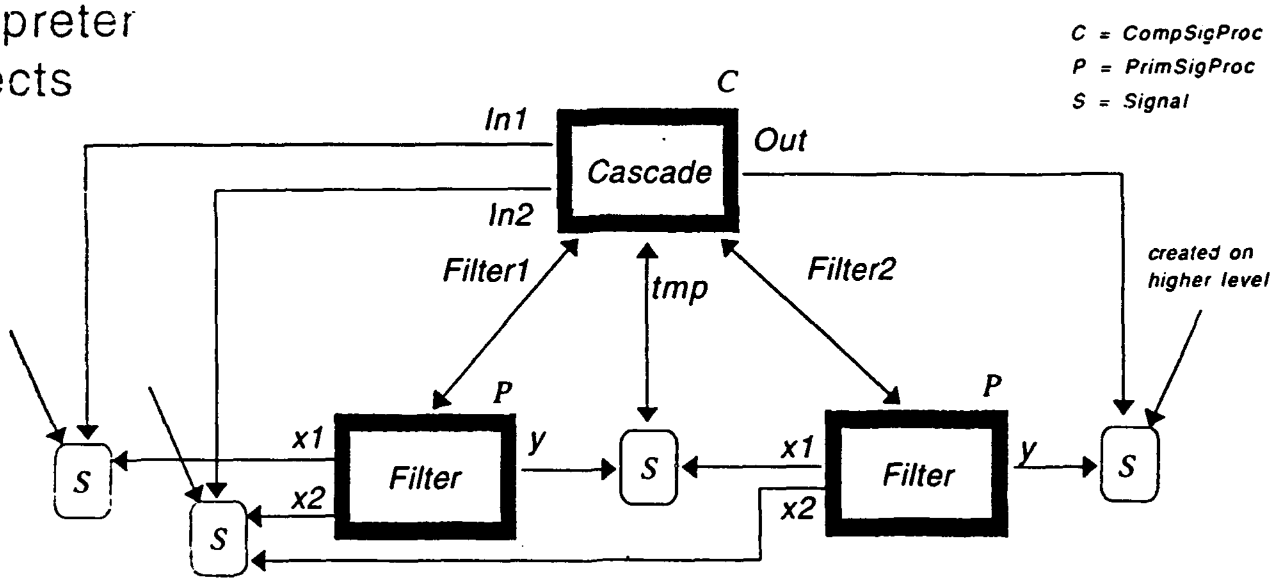

SCENARIO # 1
SCENARIO # 2
SCENARIO #3

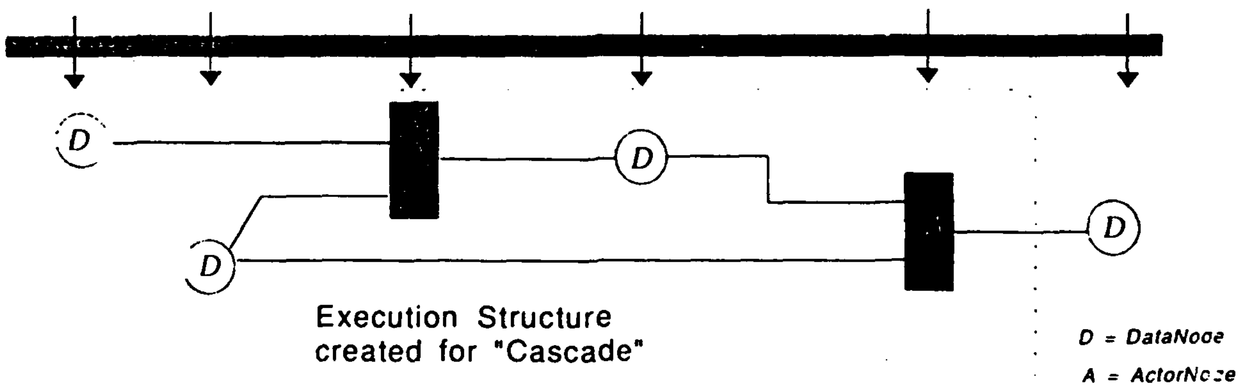
Model Interpretation

Models



Interpreter Objects





Execution Structure
created for "Cascade"

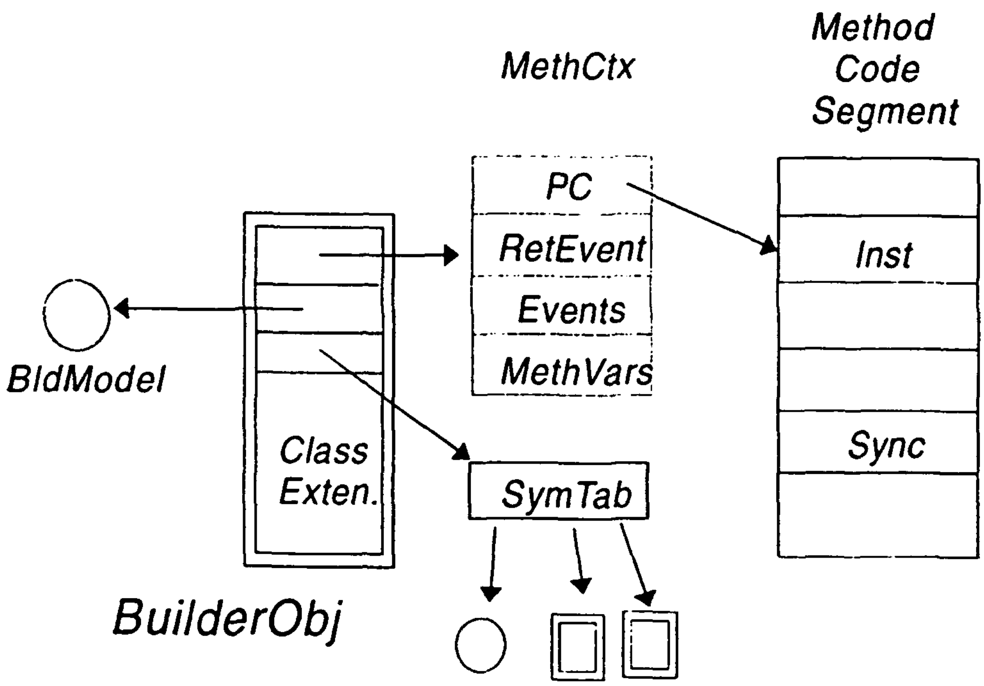
Execution Structure

Interpretation Methods

Use of High Level Operations

Compound::*Build*
{
    locals.*Instantiate*();
    subprocs.*Build*(0);
    Sync(0);
}

Primitive::Build
{
    inputs.*Resolve*(0);
    outputs.*Resolve*(0);
    Sync(0);
    MapScript();
}



Questions or comments on content should be directed to:

Dr. Janos Sztipanovits
Dept. of Electrical Engineering
Vanderbilt University
P.O. Box 1824, Station B
Nashville, TN 37235
(615) 322-3455

Or to:

Jerry Pixton
Software Productivity Consortium
2214 Rock Hill Road
Herndon, VA 22070
(703) 742-7112

Send feedback on the Consortium's Video Program and orders for video products to:

Technology Transfer Clearinghouse
Software Productivity Consortium
2214 Rock Hill Road
Herndon, VA 22070
(703) 742-7211